使用者指南

AWS 付款密碼編譯

AWS 付款密碼編譯: 使用者指南

Table of Contents

什麼是AWS付款密碼學？

AWS付款密碼學是受管理的AWS此服務可讓您存取符合支付卡產業 (PCI) 標準的付款處理所使用的密碼編譯功能和金鑰管理，而不需要您購買專用的付款 HSM 執行個體。AWSPayment Cryptography 為執行付款功能的客戶提供執行付款功能的客戶，例如收單機構、支付協助者、網路、交換器、處理器和銀行，能夠將其付款加密作業移到更接近雲端應用程式的位置，並將對包含專用支付 HSM 的輔助資料中心或託管設施的依賴降到最低。

此服務的設計是為了符合適用的產業規則，包括 PCI PIN、PCI P2PE 和 PCI DSS，而且該服務會運用原本的硬體PCI 管理工具管理協定 (HSM) V3 和 FIPS 140-2 第 3 級認證。它旨在支持低延遲和高水平的正常運行時間和彈性。AWS付款密碼技術具有完全彈性，可消除內部部署 HSM 的許多操作需求，例如佈建硬體、安全管理金鑰材料，以及在安全設施中維護緊急備份的需求。AWS付款密碼學還為您提供了以電子方式與合作夥伴共享密鑰的選項，從而無需共享紙質純文本組件。

您可以使用AWS支付密碼學控制平面 API建立和管理金鑰。

您可以使用AWS支付加密數據平面 API使用加密金鑰進行付款相關交易處理及相關的加密操作。

AWS付款密碼編譯提供重要功能，您可以使用這些功能來管理您的金鑰：

- 建立和管理對稱與非對稱AWS付款密鑰，包括 TDES，AES 和 RSA 密鑰，並指定其預期目的，例如 CVV 生成或 DUKPT 密鑰派生。
- 自動儲存您的AWS安全付款密碼編譯金鑰，受硬體安全模組 (HSM) 保護，同時在使用案例之間強制執行金鑰分離。
- 建立、刪除、列出和更新別名，這些別名是「易記名稱」，可用來存取或控制您對您的存取AWS付款密碼編譯密鑰。
- 標記您的AWS用於識別、分組、自動化、存取控制和成本追蹤的付款密鑰。
- 在之間匯入和匯出對稱金鑰AWS在 TR-31（可互操作的安全金鑰交換金鑰區塊規格）之後使用金鑰加密金鑰 (KEK) 進行付款加密和您的 HSM（或第三方）。
- 匯入和匯出之間的對稱金鑰加密金鑰 (KEK)AWS支付密碼學和其他使用非對稱密鑰對的系統，通過使用電子方式如 TR-34（使用非對稱技術分發對稱密鑰的方法）。

您可以使用AWS密碼編譯作業中的付款密碼編譯金鑰，例如：

- 以對稱或非對稱方式加密、解密及重新加密資料AWS付款密碼編譯密鑰。
- 在加密金鑰之間安全地翻譯敏感資料 (例如持卡人 PIN 碼)，而不會依照 PCI PIN 規則公開純文字。

- 生成或驗證持卡人數據，例如 CVV，CVV2 或 ARQC。
- 產生並驗證持卡人 PIN 碼。
- 產生或驗證 MAC 簽名。

概念

了解 AWS 付款密碼學中使用的基本術語和概念，以及如何使用它們來幫助您保護數據。

別名

與 AWS 付款密碼編譯金鑰相關聯的使用者易記名稱。在許多 AWS 付款密碼編譯 API 作業中，別名可與金鑰 ARN 互換使用。別名允許旋轉或以其他方式更改密鑰，而不會影響您的應用程序代碼。別名名稱是最多 256 個字元的字串。它可唯一識別帳戶和區域內相關聯的 AWS 付款密碼編譯金鑰。在 AWS 付款密碼學中，別名一律以`alias/`開頭。

別名的格式如下：

```
alias/<alias-name>
```

例如：

```
alias/sampleAlias2
```

金鑰 ARN

關鍵 ARN 是 AWS 支付密碼學中密鑰條目的 Amazon 資源名稱（ARN）。它是 AWS 付款密碼編譯金鑰的唯一、完全合格的識別碼。一種關鍵 ARN 包含一個 AWS 帳戶、區域及一隨機產生的 ID。ARN 與關鍵材料不相關或衍生出來。由於在建立或匯入作業期間會自動指派這些值，因此這些值不是冪等的。多次匯入相同的金鑰會產生多個金鑰 ARN，並具有自己的生命週期。

金鑰 ARN 的格式如下：

```
arn:<partition>:payment-cryptography:<region>:<account-id>:alias/<alias-name>
```

以下是 ARN 金鑰範例：

```
arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h
```

金鑰識別碼

密鑰標識符是對密鑰的參考，其中一個（或多個）是 AWS 支付密碼操作的典型輸入。有效的密鑰標識符可以是密鑰 Arn 或密鑰別名。

AWS 付款密碼編譯金鑰

AWS 支付密碼編譯密鑰（密鑰）用於所有加密功能。您可以使用 create key 指令直接產生金鑰，也可以透過呼叫金鑰匯入將金鑰新增至系統。金鑰的來源可以透過檢視屬性來決定 KeyOrigin。AWS 支付密碼學還支持在加密操作期間使用的派生或中間密鑰，例如 DUKPT 使用的密鑰。

這些鍵在創建時定義了不可變和可變的屬性。屬性 (例如演算法、長度和用法) 是在建立時定義的，且無法變更。其他，例如生效日期或到期日，可以修改。如需AWS 付款密碼編譯金鑰屬性的完整清單，請參閱 AWS 付款密碼編譯 API 參考資料。

AWS 付款密碼編譯金鑰具有金鑰類型，主要由 ANSI X9 TR 31 定義，這些金鑰類型主要由 ANSI X9 TR 31 定義，將其使用限制在 PCI PIN 第 3.1 版要求 19 中所指定。

當儲存、與其他帳戶共用，或按照 PCI PIN v3.1 要求 18-3 中的指定匯出時，屬性會繫結至使用金鑰區塊的金鑰。

在 AWS 付款密碼編譯平台中使用稱為金鑰 Amazon 資源名稱 (ARN) 的唯一值來識別金鑰。

> **Note**
>
> 金鑰ARN會在初始建立金鑰或匯入 AWS 付款密碼編譯服務時產生。因此，如果使用匯入金鑰功能多次新增相同的金鑰材料，相同的金鑰材料將位於多個金鑰下，但每個金鑰的生命週期都不同。

產業術語

主題

常用金鑰類型

AWK

收單機構工作金鑰 (AWK) 是一種金鑰，通常用於在收單/收單方處理器與網路 (例如 Visa 或萬事達卡) 之間交換資料。從歷史上看，AWK 利用 3DES 進行加密，並將表示為 TR31_P0_PIN 加密密鑰。

BDK

基本派生密鑰（BDK）是用於導出後續密鑰的工作密鑰，通常用作 PCI PIN 和 PCI P2PE DUKPT 過程的一部分。它被表示為基本派生密鑰。

CMK

卡主金鑰 (CMK) 是一或多個特定於卡片的金鑰，通常衍生自發卡機構主要金鑰 PAN 和 PSN，通常是 3 DES 金鑰。在個人化期間，這些金鑰會儲存在 EMV 晶片上。CMK 的範例包括交流電、SMI 和 SMC 金鑰。

厘米交流

應用程式密碼 (AC) 金鑰用作 EMV 交易的一部分，以產生交易密碼，是卡片主金鑰的一種類型。

CMK-SMI

使用安全訊息完整性 (SMI) 金鑰做為 EMV 的一部分，以驗證使用 MAC 傳送至卡片的承載的完整性，例如針腳更新指令碼。它是一種卡主密鑰。

厘米積電

安全訊息傳送機密性 (SMC) 金鑰用作 EMV 的一部分，以加密傳送至卡片的資料，例如 PIN 碼更新。它是一種卡主密鑰。

CVK

卡片驗證金鑰 (CVK) 是使用定義的演算法產生 CVV、CVV2 和類似值以及驗證輸入的金鑰。它被表示為一個 TR 31 _ 卡驗證密鑰。

IMK

發行者主要金鑰 (IMK) 是用作 EMV 晶片卡個人化一部分的主金鑰。通常會有 3 個 IMK-AC（加密），SMI（用於完整性/簽名的腳本主密鑰）和 SMC（用於機密/加密的腳本主密鑰）各一個。

IK

初始金鑰 (IK) 是 DUKPT 程序中使用的第一個金鑰，並衍生自基礎衍生金鑰 (BDK)。這個金鑰不會處理任何交易，但它會用來衍生將用於交易的 future 金鑰。建立 IK 的衍生方法是在 X9. 24:2017

中定義的。使用 TDES BDK 時，X9. 24-1:2009 是適用的標準，IK 會被初始密碼加密金鑰 (IPEK) 取代。

伊佩克

初始 PIN 碼加密金鑰 (IPEK) 是 DUKPT 程序中使用的初始金鑰，並衍生自基礎衍生金鑰 (BDK)。這個金鑰不會處理任何交易，但它會用來衍生將用於交易的 future 金鑰。IPEK 是用詞不當，因為此密鑰也可用於導出數據加密和 mac 密鑰。用於創建 IPEK 的派生方法是在 X9 中定義的。使用 AES BDK 時，X9. 24-1:2017 是適用的標準，IPEK 會被初始金鑰 (IK) 取代。

IWK

發卡機構工作金鑰 (IWK) 是一種金鑰，通常用於在發行機構/發行機構處理者與網路 (例如 Visa 或萬事達卡) 之間交換資料。從歷史上看，IWK 利用 3DES 進行加密，並以 TR31_P0_PIN 加密密鑰表示。

小桶

金鑰加密金鑰 (KEK) 是用來加密其他金鑰以進行傳輸或儲存的金鑰。根據標準，用於保護其他密鑰 KeyUsage的密鑰通常具有 TR31_K0_KEY_加密 _ 密鑰。TR-31

PEK

PIN 加密金鑰 (PEK) 是一種工作金鑰，用於加密 PIN 碼，以便在雙方之間進行儲存或傳輸。IWK 和 AWK 是 PIN 碼加密金鑰特定用途的兩個範例。這些金鑰會表示為 TR31 _P0_PIN 加密金鑰。

PGK

PGK（PIN 碼產生金鑰）是 PIN 碼驗證金鑰的另一個名稱。它實際上並不是用來產生接腳 (依預設為密碼編譯隨機數字)，而是用來產生驗證值，例如 PVV。

PVK

PIN 驗證金鑰 (PVK) 是一種用來產生 PIN 碼驗證值 (例如 PVV) 的工作金鑰。這兩種最常見的種類是用於產生 IBM3624 偏移值的 TR31_V1_IBM3624_ 驗證碼，以及用於簽證/ABA 驗證值的驗證金鑰。這也稱為「接腳產生金鑰」。

其他條款

ARQC

授權請求加密（ARQC）是 EMV 標準芯片卡（或同等的非接觸式實現）在交易時生成的密碼編譯。通常，ARQC 是由芯片卡生成的，並轉發給發行商或其代理商，以便在交易時進行驗證。

CVV

卡片驗證值是一種靜態機密值，傳統上嵌入磁條上，用於驗證交易的真實性。該算法也用於其他目的，如 ICVV，CAVV，CVV2。對於其他用例，它可能不會以這種方式引入。

CVV2

信用卡驗證值 2 是一種靜態的機密值，傳統上印在支付卡的正面（或背面），用於驗證不在場的信用卡付款（例如在電話或在線上）的真實性。它使用與 CVV 相同的算法，但服務代碼設置為 000。

智能病毒

ICVV 是類似 CV2 的值，但嵌入了 EMV（晶片）卡上的軌跡 2 等效資料。此值是使用 999 的服務代碼計算，與 CVV1/CVV2 不同，以防止竊取的資訊被用來建立不同類型的新付款憑證。例如，如果取得晶片交易資料，就無法使用此資料產生磁條 (CVV1) 或線上購買 (CVV2)。

它使用一個???密鑰

DUKPT

衍生每筆交易的唯一金鑰 (DUKPT) 是一種金鑰管理標準，通常用於定義在實體 PO/POI 上使用一次性加密金鑰的使用。從歷史上看，DUKPT 利用 3DES 進行加密。DUKPT 的業界標準定義於二零一七年十二月三日。

EMV

EMV（原來是 Europay，萬事達卡，Visa）是一個技術機構，與支付利益相關者合作創建可互操作的支付標準和技術。一個例子標準是用於芯片/非接觸式卡以及與之交互的支付終端機，包括使用的加密技術。EMV 密鑰派生是指根據初始密鑰集（例如，為每個支付卡生成唯一密鑰的方法）IMK

HSM

硬體安全模組 (HSM) 是一種實體裝置，可保護密碼編譯作業 (例如，加密、解密和數位簽章)，以及用於這些作業的基礎金鑰。

KCV

索引鍵檢查值 (KCV) 是指各種校驗和方法，主要用來比較金鑰之間的索引鍵，而無需存取實際的金鑰資料。KCV 也被用於完整性驗證（尤其是在交換密鑰時），儘管此角色現在已包含在關鍵區塊格式中，例如. TR-31 對於 TDES 金鑰，KCV 的計算方式是將 8 個位元組加密，每個位元組的值為零，並保留加密結果的 3 個最高順序位元組。對於 AES 金鑰，KCV 是使用 CMAC 演算法計算，其中輸入資料為零的 16 個位元組，並保留加密結果的 3 個最高階位元組。

KDH

金鑰發佈主機 (KDH) 是在金鑰交換程序 (例如 TR-34) 中傳送金鑰的裝置或系統。從 AWS 支付密碼學發送密鑰時，它被認為是 KDH。

KIF

密鑰注入工具（KIF）是用於初始化支付終端機（包括使用加密密鑰加載它們）的安全設施。

KRD

金鑰接收裝置 (KRD) 是在金鑰交換程序 (例如 TR-34) 中接收金鑰的裝置。當發送密鑰 AWS 支付密碼學，它被認為是 KRD。

KSN

密鑰序列號（KSN）是用作 DUKPT 加密/解密的輸入值，以便在每個交易中創建唯一的加密密鑰。KSN 通常由 BDK 標識符，半唯一終端 ID 以及一個事務計數器組成，該計數器會在給定支付終端上處理的每個轉換時遞增。

mPOC

mPoC (商用硬體上的行動銷售點) 是 PCI 標準，可滿足商家使用智慧型手機或其他商業 off-the-shelf (COTS) 行動裝置接受持卡人 PIN 碼或非接觸式付款的解決方案的安全要求。

鍋

「主要帳戶號碼」(PAN) 是信用卡或扣帳卡等帳戶的唯一識別碼。通常長度為 13-19 位數字。前 6-8 位數字識別網絡和發卡銀行。

針腳擋塊

在處理或傳輸過程中包含 PIN 以及其他數據元素的數據塊。PIN 圖塊格式標準化 PIN 圖塊的內容，以及如何處理以擷取 PIN 碼。大多數 PIN 塊由 PIN 碼，PIN 長度組成，並且通常包含 PAN 的部分或全部。 AWS 支付密碼技術支持 ISO 9564-1 格式 0，1，3 和 4。AES 金鑰需要格式 4。驗證或轉譯 PIN 時，需要指定傳入或傳出資料的 PIN 碼區塊。

POI

互動點（POI）也經常與銷售點（POS）同義使用，是持卡人與其互動以顯示其付款憑據的硬件設備。POI 的一個例子是商家位置的實體終端機。如需獲得認證的 PCI PTS POI 終端機清單，請參閱 PCI 網站。

PSN

PAN 序號（PSN）是用於區分使用相同 PAN 發行的多張卡的數值。

公有金鑰

使用非對稱密碼 (RSA) 時，公開金鑰是公開-私密 key pair 組的公開元件。公有金鑰可以共用並分配至需要為公有-私有金鑰對擁有者加密資料的實體。對於數位簽章操作，公有金鑰用於驗證簽章。

私有金鑰

使用非對稱密碼 (RSA) 時，私密金鑰是公用-私密金鑰組的私密元件。私有金鑰用於解密資料或建立數位簽章。與對稱 AWS 式付款密碼編譯金鑰類似，私密金鑰是由 HSM 安全建立的。它們只會解密到 HSM 的揮發性記憶體中，而且只會在處理加密要求所需的時間內解密。

PVV

PIN 驗證值 (PVV) 是一種密碼編譯輸出類型，可用於在不儲存實際接腳的情況下驗證引腳。雖然它是一個通用術語，但在 AWS 付款密碼學的背景下，PVV 是指 Visa 或 ABA PVV 方法。這個 PVV 是一個四位數的數字，其輸入包括卡號，平移序列號，平移本身和 PIN 驗證密鑰。在驗證階段期間， AWS 付款密碼編譯會使用交易資料在內部重新建立 PVV，並再次比較 AWS 付款密碼編譯客戶所儲存的值。通過這種方式，它在概念上類似於加密哈希或 MAC。

RSA 包裝/展開包裝

RSA 換行使用非對稱金鑰來包裝對稱金鑰 (例如 TDES 金鑰) 以傳輸到另一個系統。只有具有相符私密金鑰的系統才能解密承載並載入對稱金鑰。相反，RSA 解除包裝，將安全地解密使用 RSA 加密的密鑰，然後將該密鑰加載到支付密碼學中。 AWS RSA wrap 是一種交換密鑰的低級方法，不以密鑰塊格式傳輸密鑰，並且不使用發送方的有效負載簽名。應考慮備用控制以確定天意和關鍵屬性沒有突變。

TR-34 也在內部使用 RSA，但它是一種單獨的格式，不可互操作。

TR-31

TR-31（正式定義為 ANSI X9 TR 31）是由美國國家標準協會（ANSI）定義的關鍵區塊格式，用於支援在與關鍵資料本身相同的資料結構中定義關鍵屬性。TR-31 鍵圖塊格式定義了一組關鍵屬性，這些屬性與鍵相關聯，以便將它們保存在一起。 AWS 付款密碼學盡可能使用 TR-31 標準化術語，以確保適當的密鑰分離和密鑰目的。TR-31 已被二年十四月二十四日取代。

TR-34

TR-34 是 ANSI X9.24-2 的一項實作，其中描述了使用非對稱技術 (例如 RSA) 安全散佈對稱金鑰 (例如 3DES 和 AES) 的通訊協定。 AWS 付款密碼學使用 TR-34 方法來允許密鑰的安全導入和導出。

相關服務

AWS Key Management Service

AWS金鑰管理服務 (AWSKMS) 是一項受管理服務，可讓您輕鬆建立和控制用來保護資料的加密金鑰。AWSKMS 使用硬體安全模組 (HSM) 來保護和驗證您的AWSKMS 金鑰。

AWS CloudHSM

AWS CloudHSM為客戶提供專用的一般用途 HSM 執行個體AWS雲端。AWS CloudHSM可以提供各種加密功能，例如創建密鑰，數據簽名或加密和解密數據。

如需詳細資訊

- 若要瞭解中使用的術語和概念AWS付款密碼學，請參閱AWS付款密碼學概念。
- 如需有關的資訊AWS支付密碼編譯控制平面 API，請參閱AWS支付密碼學控制平面 API 參考。
- 如需有關的資訊AWS付款加密資料平面 API，請參閱AWS付款密碼學資料平面 API 參考。
- 有關如何進行的詳細技術信息AWS支付密碼學使用密碼學和安全AWS付款密碼編譯金鑰，請參閱密碼詳細資料。

用於的端點 AWS Payment Cryptography

若要 AWS Payment Cryptography以程式設計方式連線到，請使用服務進入點的端點。URL AWS SDKs和命令列工具會 AWS 區域 根據要求的區域內容，自動使用服務的預設端點，因此通常不需要明確設定這些值。必要時，您可以為API請求指定不同的端點。

控制平面端點

區域名稱	區域	端點	通訊協定
美國東部 (維吉尼亞北部)	us-east-1	控制計劃. 支付加密.	HTTPS
美國東部 (俄亥俄)	us-east-2	控制計劃. 支付加密.	HTTPS

區域名稱	區域	端點	通訊協定
美國西部 (奧勒岡)	us-west-2	控制計劃. 支付加密. 我們西部-2.	HTTPS
亞太區域 (新加坡)	ap-southeast-1	控制計劃. 支付加密. AP-東南部	HTTPS
亞太區域 (東京)	ap-northeast-1	控制計劃. 支付加密. AP-東北-1.	HTTPS
歐洲 (法蘭克福)	eu-central-1	控制計劃。支付加密。歐盟-中央-1.amazonaws.com	HTTPS
歐洲 (愛爾蘭)	eu-west-1	控制計劃。支付加密。歐盟西部-亞馬遜	HTTPS

資料平面端點

區域名稱	區域	端點	通訊協定
美國東部 (維吉尼亞北部)	us-east-1	數據支付加密。我們東部-	HTTPS
美國東部 (俄亥俄)	us-east-2	數據支付加密。我們東部-亞馬遜	HTTPS
美國西部 (奧勒岡)	us-west-2	數據支付加密. 我們西部-2.	HTTPS
亞太區域 (新加坡)	ap-southeast-1	數據支付加密. AP-東南部-亞馬遜	HTTPS

區域名稱	區域	端點	通訊協定
亞太區域 (東京)	ap-northeast-1	支付密碼. 支付加密. AP-東北-1.	HTTPS
歐洲 (法蘭克福)	eu-central-1	支付加密。歐盟中央-1. 亞馬遜	HTTPS
歐洲 (愛爾蘭)	eu-west-1	支付加密。歐盟西部-亞馬遜	HTTPS

開始使用 AWS 付款密碼

要開始使用 AWS 付款密碼學，您首先要創建密鑰，然後在各種加密操作中使用它們。下面的教程提供了生成用於生成/驗證CVV2值的密鑰的簡單用例。要嘗試其他示例並探索其中的部署模式AWS，請嘗試以下AWS 付款密碼學研討會或瀏覽 Github 上提供的示例項目

本教學課程將逐步引導您建立單一金鑰，以及使用金鑰執行密碼編譯作業。之後，如果您不再需要，請刪除該金鑰，這樣就會完成金鑰生命週期。

> **Warning**
>
> 本使用者指南中的範例可能會使用範例值。我們強烈建議您不要在生產環境中使用樣本值，例如關鍵序號。

主題

必要條件

在開始之前，請確保：

- 您有權訪問該服務。如需詳細資訊，請參閱 IAM 政策。
- 您已經安AWS CLI裝了。您也可以使用AWS SDKs或存AWS APIs取 AWS 付款密碼，但本教學課程中的指示會使用 AWS CLI.

步驟 1：建立金鑰

第一步是創建一個密鑰。在本自學課程中，您會建立CVK雙長度 3 DES (2 KEYTDES) 鍵來產生和驗證CVV/CVV2值。

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
        "KeyAttributes": {
            "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyAlgorithm": "TDES_2KEY",
            "KeyModesOfUse": {
                "Encrypt": false,
                "Decrypt": false,
                "Wrap": false,
                "Unwrap": false,
                "Generate": true,
                "Sign": false,
                "Verify": true,
                "DeriveKey": false,
                "NoRestrictions": false
            }
        },
        "KeyCheckValue": "CADDA1",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "Enabled": true,
        "Exportable": true,
        "KeyState": "CREATE_COMPLETE",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
        "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
    }
}
```

請注意代表密鑰的，例如 ARN：AW：支付密碼：美國東部 2：11112222333：密鑰/tqv5yij6w txx64pi。`KeyArn`您需要在下一步中。

步驟 2：使用密鑰生成CVV2值

在此步驟中，您可以使用步驟 1 中的金鑰CVV2為指定日期PAN和到期日產生。

```
$  aws payment-cryptography-data generate-card-validation-data \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi \
    --primary-account-number=171234567890123 \
    --generation-attributes CardVerificationValue2={CardExpiryDate=0123}
```

```
{
    "CardDataGenerationKeyCheckValue": "CADDA1",
    "CardDataGenerationKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/tqv5yij6wtxx64pi",
    "CardDataType": "CARD_VERIFICATION_VALUE_2",
    "CardDataValue": "144"
}
```

請注意`cardDataValue`，在這種情況下是 3 位數字 144。您需要在下一步中。

步驟 3：驗證步驟 2 中產生的值

在此範例中，您可以使用您在步驟 1 中建立的金鑰CVV2來驗證步驟 2。

執行下列命令來驗證CVV2.

```
$  aws payment-cryptography-data  verify-card-validation-data \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi \
    --primary-account-number=171234567890123 \
    --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
    --validation-data 144
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "CADDA1"
}
```

服務會傳HTTP回 200 的回應，以指出已驗證CVV2。

步驟 4：執行陰性測試

在此步驟中，您會建立不正確CVV2且未驗證的負面測試。您嘗試CVV2使用您在步驟 1 中建立的金鑰來驗證不正確的項目。例如，如果持卡人CVV2在結帳時輸入錯誤，則這是預期的操作。

```
$  aws payment-cryptography-data  verify-card-validation-data \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi \
    --primary-account-number=171234567890123 \
    --verification-attributes CardVerificationValue2={CardExpiryDate=0123} \
    --validation-data 999
```

```
Card validation data verification failed.
```

該服務返HTTP回 400 的響應消息「卡驗證數據驗證失敗」和 INVALID VALIDATION _ 的原因DATA。

步驟 5：（可選）清理

現在，您可以刪除在步驟 1 中創建的密鑰。為了將無法復原的變更降到最低，預設的金鑰刪除期間為七天。

```
$ aws payment-cryptography delete-key \
    --key-identifier=arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi
```

```
{
    "Key": {
        "CreateTimestamp": "2022-10-27T08:27:51.795000-07:00",
        "DeletePendingTimestamp": "2022-11-03T13:37:12.114000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_3KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": true,
                "DeriveKey": false,
                "Encrypt": true,
```

```
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": true,
                "Verify": false,
                "Wrap": true
            },
            "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY"
        },
        "KeyCheckValue": "CADDA1",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "DELETE_PENDING",
        "UsageStartTimestamp": "2022-10-27T08:27:51.753000-07:00"
    }
}
```

注意輸出中的兩個字段。依`deletePendingTimestamp`預設，會設定為 future 七天。設 keyState 定為`DELETE_PENDING`。您可以在排定的刪除時間之前隨時透過電話取消此刪除`restore-key`。

管理金鑰

若要開始使用 AWS 付款密碼編譯，您需要建立 AWS 付款密碼編譯金鑰。

本節中的主題說明如何建立和管理從建立到刪除的各種 AWS 付款密碼編譯金鑰類型。它包含建立、編輯和檢視金鑰、標記金鑰、建立金鑰別名，以及啟用和停用金鑰等主題。

主題

產生金鑰

您可以使用 操作建立 AWS 付款密碼編譯金鑰 CreateKey API。在此過程中，您將指定金鑰或結果輸出的各種屬性，例如金鑰演算法（例如 TDES_3KEY）、 KeyUsage（例如 TR31_P0_PINENCRYPTION__KEY）、允許的操作（例如加密、簽署），以及是否可以匯出。建立 AWS 付款密碼編譯金鑰後，您無法變更這些屬性。

範例

- 產生 2KEY TDES金鑰
- 產生 Pin 加密金鑰
- 建立非對稱（RSA）金鑰
- 產生PIN驗證值（PVV）金鑰
- 產生非對稱ECC金鑰

產生 2KEY TDES金鑰

Example

此命令會產生 2KEY 個TDES金鑰，用於產生和驗證 CVV/CVV2 值。回應會回傳請求參數，包括 ARN 用於後續呼叫，以及 KCV（金鑰檢查值）。

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,\
KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY, \
KeyModesOfUse='{Generate=true,Verify=true}'
```

```
                        {
    "Key": {
        "CreateTimestamp": "2022-10-26T16:04:11.642000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
hjprdg5o4jtgs5tw",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_2KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": false,
                "Encrypt": false,
                "Generate": true,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": false,
                "Verify": true,
                "Wrap": false
            },
            "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY"
        },
        "KeyCheckValue": "B72F",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-26T16:04:11.559000-07:00"
    }
```

```
}
```

產生 Pin 加密金鑰

Example 產生 Pin 加密金鑰（PEK）

此命令會產生 3KEY 個TDES金鑰，用於加密PIN值（稱為 Pin Encryption Key）。此金鑰可用於保護儲存，PINs或在驗證嘗試期間PINs提供的解密，例如交易期間。回應會回傳請求參數，包括 ARN用於後續呼叫，以及 KCV（金鑰檢查值）。

```
$ aws payment-cryptography create-key --exportable --key-attributes \
                        KeyAlgorithm=TDES_3KEY,KeyUsage=TR31_P0_PIN_ENCRYPTION_KEY, \
                        KeyClass=SYMMETRIC_KEY,/

 KeyModesOfUse='{Encrypt=true,Decrypt=true,Wrap=true,Unwrap=true}'
```

```
{
    "Key": {
        "CreateTimestamp": "2022-10-27T08:27:51.795000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_3KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": true,
                "DeriveKey": false,
                "Encrypt": true,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": true,
                "Verify": false,
                "Wrap": true
            },
            "KeyUsage": "TR31_P0_PIN_ENCRYPTION_KEY"
        },
```

```
        "KeyCheckValue": "9CA6",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-27T08:27:51.753000-07:00"
    }
}
```

建立非對稱（RSA）金鑰

Example

在此範例中，我們將產生新的非對稱 RSA 2048 位元金鑰對。將產生新的私有金鑰以及相符的公有金鑰。可以使用 getPublicCertificate 擷取公有金鑰API。

```
$ aws payment-cryptography create-key --exportable \
--key-attributes
 KeyAlgorithm=RSA_2048,KeyUsage=TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION, \
KeyClass=ASYMMETRIC_KEY_PAIR,KeyModesOfUse='{Encrypt=true,
 Decrypt=True,Wrap=True,Unwrap=True}'
```

```
{
    "Key": {
        "CreateTimestamp": "2022-11-15T11:15:42.358000-08:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
nsq2i3mbg6sn775f",
        "KeyAttributes": {
            "KeyAlgorithm": "RSA_2048",
            "KeyClass": "ASYMMETRIC_KEY_PAIR",
            "KeyModesOfUse": {
                "Decrypt": true,
                "DeriveKey": false,
                "Encrypt": true,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": true,
                "Verify": false,
```

```
                "Wrap": true
            },
            "KeyUsage": "TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION"
        },
        "KeyCheckValue": "40AD487F",
        "KeyCheckValueAlgorithm": "CMAC",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-11-15T11:15:42.182000-08:00"
    }
}
```

產生PIN驗證值（PVV）金鑰

Example

此命令會產生 3KEY 個TDES金鑰，用於產生PVV值（稱為 Pin 驗證值）。您可以使用此金鑰來產生可與後續計算的 進行比較PVV的值PVV。回應會回傳請求參數，包括 ARN用於後續呼叫，以及 KCV（金鑰檢查值）。

```
$ aws payment-cryptography create-key --exportable/
--key-attributes KeyAlgorithm=TDES_3KEY,KeyUsage=TR31_V2_VISA_PIN_VERIFICATION_KEY,/
KeyClass=SYMMETRIC_KEY,KeyModesOfUse='{Generate=true,Verify=true}'
```

```
{
    "Key": {
        "CreateTimestamp": "2022-10-27T10:22:59.668000-07:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
j4u4cmnzkelhc6yb",
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_3KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": false,
                "Encrypt": false,
                "Generate": true,
                "NoRestrictions": false,
                "Sign": false,
```

```
                "Unwrap": false,
                "Verify": true,
                "Wrap": false
            },
            "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY"
        },
        "KeyCheckValue": "5132",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2022-10-27T10:22:59.614000-07:00"
    }
}
```

產生非對稱ECC金鑰

Example

此命令會產生ECC金鑰對，目的是透過從其公有私有金鑰對中取得共用金鑰，在兩方之間建立ECDH（Elliptic Curve Diffie-Hellman） 金鑰協議。使用 ECDH時，每一方都會產生自己的ECC金鑰對，其中包含金鑰用途 K3 和使用 X 的模式，並交換公有金鑰。雙方接著會使用其私有金鑰和收到的公有金鑰來建立共用衍生金鑰。

為了維持付款中密碼編譯金鑰的單次使用原則，建議不要重複使用ECC金鑰對進行多種用途，例如ECDH金鑰衍生和簽署。

```
$ aws payment-cryptography create-key --exportable/
--key-attributes
 KeyAlgorithm=ECC_NIST_P256,KeyUsage=TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT,/
KeyClass=ASYMMETRIC_KEY_PAIR,KeyModesOfUse='{DeriveKey=true}'
```

```
{
    "Key": {
        "CreateTimestamp": "2024-10-17T01:31:55.908000+00:00",
        "Enabled": true,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-west-2:075556953750:key/
xzydvquw6ejfxnwq",
```

```
        "KeyAttributes": {
            "KeyAlgorithm": "ECC_NIST_P256",
            "KeyClass": "ASYMMETRIC_KEY_PAIR",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": true,
                "Encrypt": false,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": false,
                "Verify": false,
                "Wrap": false
            },
            "KeyUsage": "TR31_K3_ASYMMETRIC_KEY_FOR_KEY_AGREEMENT"
        },
        "KeyCheckValue": "7E34F19F",
        "KeyCheckValueAlgorithm": "CMAC",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2024-10-17T01:31:55.866000+00:00"
    }
}
```

列出金鑰

List Keys 會顯示此帳戶和區域中來電者可以存取的金鑰清單。

Example

```
$ aws payment-cryptography list-keys
```

```
 {"Keys": [
    {
        "CreateTimestamp": "2022-10-12T10:58:28.920000-07:00",
        "Enabled": false,
        "Exportable": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
alsuwfxug3pgy6xh",
```

```
        "KeyAttributes": {
            "KeyAlgorithm": "TDES_3KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": true,
                "DeriveKey": false,
                "Encrypt": true,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": true,
                "Verify": false,
                "Wrap": true
            },
            "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
        },
        "KeyCheckValue": "369D",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStopTimestamp": "2022-10-27T14:19:42.488000-07:00"
    }
    ]
}
```

啟用和停用金鑰

您可以停用並重新啟用 AWS 付款密碼編譯金鑰。建立金鑰時，預設會啟用該金鑰。如果您停用金鑰，則在您重新啟用金鑰之前，無法在任何密碼編譯操作中使用金鑰。啟動/停止用量命令會立即生效，因此建議您在進行此類變更之前檢閱用量。您也可以使用選用`timestamp`參數，將變更（開始或停止用量）設定為在未來生效。

由於它是暫時且容易復原的，因此停用 AWS Payment Cryptography 金鑰是刪除 AWS Payment Cryptography 金鑰更安全的替代方案，這是具有破壞性和不可復原性的動作。如果您正在考慮刪除 AWS 付款密碼編譯金鑰，請先將其停用，並確保未來不需要使用金鑰來加密或解密資料。

主題

- 開始金鑰用量
- 停止金鑰用量

開始金鑰用量

必須啟用金鑰用量，才能將金鑰用於密碼編譯操作。如果未啟用金鑰，您可以使用此操作使其可用。欄位`UsageStartTimestamp`代表金鑰何時變成/將變成作用中。啟用的權杖將會是過去的，如果是待啟用，則未來會是過去的權杖。

Example

在此範例中，系統會針對金鑰用量請求啟用金鑰。回應包含金鑰資訊，且啟用旗標已轉換為 true。這也會反映在 list-keys 回應物件中。

```
$ aws payment-cryptography start-key-usage --key-identifier "arn:aws:payment-
cryptography:us-east-2:111122223333:key/alsuwfxug3pgy6xh"
```

```
{
      "Key": {
      "CreateTimestamp": "2022-10-12T10:58:28.920000-07:00",
      "Enabled": true,
      "Exportable": true,
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
alsuwfxug3pgy6xh",
      "KeyAttributes": {
          "KeyAlgorithm": "TDES_3KEY",
          "KeyClass": "SYMMETRIC_KEY",
          "KeyModesOfUse": {
              "Decrypt": true,
              "DeriveKey": false,
              "Encrypt": true,
              "Generate": false,
              "NoRestrictions": false,
              "Sign": false,
              "Unwrap": true,
              "Verify": false,
              "Wrap": true
          },
          "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
      },
      "KeyCheckValue": "369D",
      "KeyCheckValueAlgorithm": "ANSI_X9_24",
      "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
      "KeyState": "CREATE_COMPLETE",
      "UsageStartTimestamp": "2022-10-27T14:09:59.468000-07:00"
```

```
  }
}
```

停止金鑰用量

如果您不再計劃使用金鑰，您可以停止金鑰使用，以防止進一步的密碼編譯操作。此操作不是永久性的，因此您可以使用啟動金鑰用量 將其反轉。您也可以將金鑰設定為未來停用。欄位`UsageStopTimestamp`代表金鑰何時變成/將變成停用。

Example

在此範例中，系統會要求 在未來停止金鑰用量。執行後，除非透過啟動金鑰用量重新啟用，否則此金鑰無法用於密碼編譯操作。回應包含金鑰資訊，且啟用旗標已轉換為 false。這也會反映在 list-keys 回應物件中。

```
$ aws payment-cryptography stop-key-usage --key-identifier "arn:aws:payment-
cryptography:us-east-2:111122223333:key/alsuwfxug3pgy6xh"
```

```
{
  "Key": {
      "CreateTimestamp": "2022-10-12T10:58:28.920000-07:00",
      "Enabled": false,
      "Exportable": true,
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
alsuwfxug3pgy6xh",
      "KeyAttributes": {
          "KeyAlgorithm": "TDES_3KEY",
          "KeyClass": "SYMMETRIC_KEY",
          "KeyModesOfUse": {
              "Decrypt": true,
              "DeriveKey": false,
              "Encrypt": true,
              "Generate": false,
              "NoRestrictions": false,
              "Sign": false,
              "Unwrap": true,
              "Verify": false,
              "Wrap": true
          },
          "KeyUsage": "TR31_P1_PIN_GENERATION_KEY"
      },
```

```
        "KeyCheckValue": "369D",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "KeyState": "CREATE_COMPLETE",
        "UsageStopTimestamp": "2022-10-27T14:09:59.468000-07:00"
    }
}
```

刪除金鑰

刪除 AWS Payment Cryptography 金鑰會刪除金鑰材料和與金鑰相關聯的所有中繼資料，除非金鑰副本在 AWS Payment Cryptography 之外可用，否則將無法復原。刪除金鑰後，您無法再解密在該金鑰下加密的資料，這表示資料可能無法復原。只有當您確定不再需要使用金鑰，且沒有其他方正在使用此金鑰時，才應該刪除金鑰。如果您不確定，請考慮停用 金鑰，而不是刪除它。如果您稍後需要再次使用已刪除的 AWS 付款密碼編譯金鑰，您可以重新啟用停用的金鑰，但除非您能夠從其他來源重新匯入，否則無法復原已刪除的付款密碼編譯金鑰。

在刪除金鑰之前，您應該確保不再需要金鑰。 AWS 付款密碼編譯不會儲存密碼編譯操作的結果，CVV2也無法判斷任何持久性密碼編譯材料是否需要金鑰。

AWS 除非您明確排定刪除金鑰，且強制等待期過期，否則付款密碼編譯永遠不會刪除屬於作用中 AWS 帳戶的金鑰。

不過，由於下列一個或多個原因，您可以選擇刪除 AWS 付款密碼編譯金鑰：

- 為不再需要的金鑰完成金鑰生命週期
- 為了避免與維護未使用的 AWS 付款密碼編譯金鑰相關聯的管理開銷

> **Note**
>
> 如果您關閉或刪除 AWS 帳戶，則無法存取您的 AWS 付款密碼編譯金鑰。您不需要將刪除 AWS 付款密碼編譯金鑰與關閉帳戶分開排程。

AWS 當您排程刪除 AWS 付款密碼編譯金鑰，以及實際刪除 AWS 付款密碼編譯金鑰時，付款密碼編譯會在AWS CloudTrail日誌中記錄項目。

關於等待期

由於刪除金鑰是不可逆的， AWS 因此付款密碼編譯需要您設定介於 3–180 天的等待期。預設等待期為七天。

不過，實際等待期可能比您排定的等待期長最多 24 小時。若要取得要刪除 AWS 付款密碼編譯金鑰的實際日期和時間，請使用 GetKey 操作。請務必注意時區。

在等待期間， AWS 付款密碼編譯金鑰狀態和金鑰狀態為待刪除。

> **Note**
>
> 待刪除的 AWS 付款密碼編譯金鑰不能用於任何密碼編譯操作 。

等待期結束後， AWS 付款密碼編譯會刪除 AWS 付款密碼編譯金鑰、其別名和所有相關 AWS 付款密碼編譯中繼資料。

使用等待期來確保您現在或未來不需要 AWS 付款密碼編譯金鑰。如果您在等待期間確實需要 金鑰，您可以在等待期間結束前取消刪除金鑰。等待期結束後，您無法取消刪除金鑰，且服務會刪除金鑰。

Example

在此範例中，系統會請求刪除金鑰。除了基本金鑰資訊之外，兩個相關欄位表示金鑰狀態已變更為 DELETE_PENDING，並 deletePendingTimestamp表示目前排程刪除金鑰的時間。

```
$ aws payment-cryptography delete-key \
                     --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h
```

```
  {
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
        "KeyAttributes": {
            "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyAlgorithm": "TDES_3KEY",
            "KeyModesOfUse": {
```

```
                "Encrypt": false,
                "Decrypt": false,
                "Wrap": false,
                "Unwrap": false,
                "Generate": true,
                "Sign": false,
                "Verify": true,
                "DeriveKey": false,
                "NoRestrictions": false
            }
        },
        "KeyCheckValue": "",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "Enabled": false,
        "Exportable": true,
        "KeyState": "DELETE_PENDING",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "CreateTimestamp": "2023-06-05T12:01:29.969000-07:00",
        "UsageStopTimestamp": "2023-06-05T14:31:13.399000-07:00",
        "DeletePendingTimestamp": "2023-06-12T14:58:32.865000-07:00"
    }
}
```

Example

在此範例中，待定刪除已取消。成功完成後，就不會再根據先前的排程刪除金鑰。回應包含基本金鑰資訊；此外，兩個相關欄位已變更 - `KeyState`和 `deletePendingTimestamp`。 `KeyState` 會傳回值 CREATE_COMPLETE，但 `DeletePendingTimestamp` 會移除。

```
$ aws payment-cryptography restore-key --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h
```

```
{
    "Key": {
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
        "KeyAttributes": {
            "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY",
            "KeyClass": "SYMMETRIC_KEY",
            "KeyAlgorithm": "TDES_3KEY",
            "KeyModesOfUse": {
```

```
                "Encrypt": false,
                "Decrypt": false,
                "Wrap": false,
                "Unwrap": false,
                "Generate": true,
                "Sign": false,
                "Verify": true,
                "DeriveKey": false,
                "NoRestrictions": false
            }
        },
        "KeyCheckValue": "",
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "Enabled": false,
        "Exportable": true,
        "KeyState": "CREATE_COMPLETE",
        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
        "CreateTimestamp": "2023-06-08T12:01:29.969000-07:00",
        "UsageStopTimestamp": "2023-06-08T14:31:13.399000-07:00"
    }
}
```

匯入和匯出金鑰

AWS 付款密碼編譯金鑰可以從其他 解決方案匯入或匯出至其他 解決方案（例如其他 HSMs）。使用匯入和匯出功能與服務供應商交換金鑰是常見的使用案例。作為雲端服務， AWS Payment Cryptography 採用現代的電子方法來管理金鑰，同時協助您維持適用的合規和控制。長期目標是從紙本型金鑰元件轉向以標準為基礎的電子金鑰交換方式。

金鑰加密金鑰（KEK）交換

AWS 付款密碼編譯鼓勵使用公有金鑰密碼編譯（RSA）進行使用成熟 ANSI X9.24 TR-34 規範的初始金鑰交換。此初始金鑰類型的一般名稱包括金鑰加密金鑰（KEK）、區域主金鑰（ZMK）和區域控制主金鑰（）ZCMK。如果您的系統或合作夥伴尚未支援 TR-34，您也可以考慮使用 RSA Wrap/Unwrap 。

如果您需要繼續處理紙本金鑰元件，直到所有合作夥伴都支援電子金鑰交換，您可以考慮HSM為此保留離線。

> **Note**
> 如果您想要匯入自己的測試金鑰，請在 Github 上查看範例專案。如需如何從其他平台匯入/匯出金鑰的指示，請參閱這些平台的使用者指南。

工作金鑰（WK）交換

AWS 付款密碼編譯使用相關的產業規範（ANSI X9.24 TR 31-2018）來交換工作金鑰。TR-31 假設 KEK 先前已交換。這符合 要求PCIPIN，一律以密碼編譯方式將金鑰材料繫結至其金鑰類型和用量。工作金鑰具有各種名稱，包括收單機構工作金鑰、發行者工作金鑰、IPEK、 BDK等。

主題

匯入金鑰

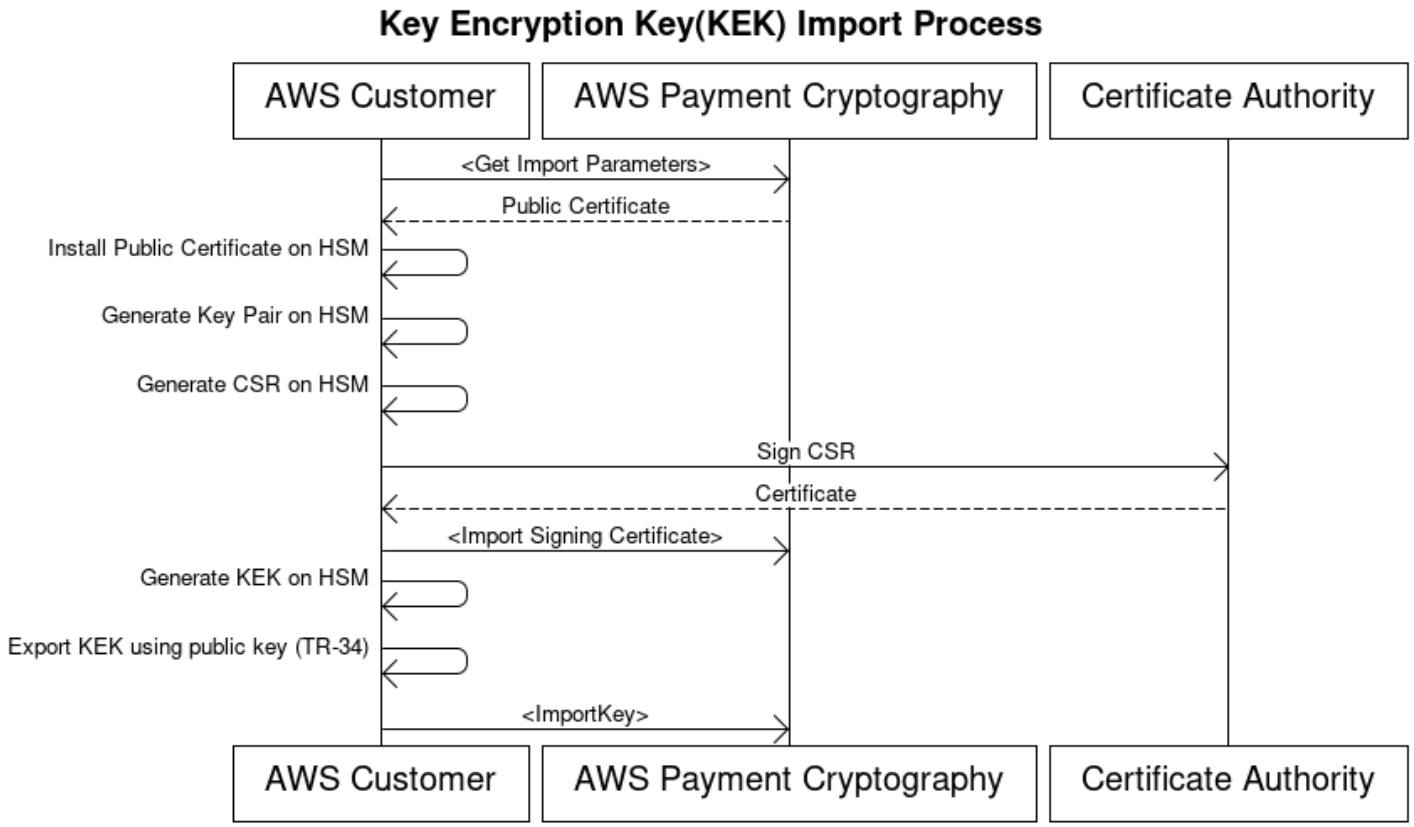
> **Important**
> 範例可能需要 VAWSCLIV2 的最新版本。開始使用之前，請確定您已升級至最新版本。

主題

匯入對稱金鑰

主題

使用非對稱技術匯入金鑰（TR-34）



概觀：TR-34 使用RSA非對稱密碼編譯來加密對稱金鑰以進行交換，並確保資料來源（簽署）。這可確保包裝金鑰的機密性（加密）和完整性（簽章）。

如果您想要匯入自己的金鑰，請在 Github 上查看範例專案。如需如何從其他平台匯入/匯出金鑰的指示，請參閱這些平台的使用者指南。

1. 呼叫初始化匯入命令

   呼叫 `get-parameters-for-import` 以初始化匯入程序。這API將產生金鑰對，用於金鑰匯入、簽署金鑰並傳回憑證和憑證根。最後，要匯出的金鑰應該使用此金鑰加密。在 TR-34 術語中，這稱為 KRD Cert。請注意，這些憑證為短期使用，僅供此用途使用。

2. 在金鑰來源系統上安裝公有憑證

   對於許多 HSMs，您可能需要存取步驟 1 中產生的install/load/trust公有憑證，才能使用它匯出金鑰。

3. 產生公有金鑰並提供 AWS 付款密碼編譯的憑證根目錄

   為了確保傳輸承載的完整性，其由傳送方簽署（稱為金鑰分發主機或 KDH）。傳送方會想要產生用於此目的的公有金鑰，然後建立可提供給 AWS Payment Cryptography 的公有金鑰憑證（X509）。 AWS Private CA 是產生憑證的一個選項，但對使用的憑證授權機構沒有限制。

   取得憑證後，您會想要使用 `importKey` 命令和 KeyMaterialType `ROOT_PUBLIC_KEY_CERTIFICATE`的 和 將根憑證載入 AWS 付款密碼編譯 KeyUsageType `TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`。

4. 從來源系統匯出金鑰

   許多 HSMs和 相關系統支援使用 TR-34 規範匯出金鑰的能力。您需要將步驟 1 的公有金鑰指定為 KRD（加密）憑證，並將步驟 3 的金鑰指定為 KDH（簽署）憑證。若要匯入至 AWS 付款密碼編譯，您需要將格式指定為 TR-34.2012 CMS非雙通格式，這些格式也可能稱為 TR-34 Diebold 格式。

5. 呼叫匯入金鑰

   作為最後一個步驟，您將使用 KeyMaterialType 的 importKey API呼叫 `TR34_KEY_BLOCK`。`certificate-authority-public-key-identifier` 將是步驟 3 中匯入根 CA 的金鑰ARN，`key-material`將包裝步驟 4 的金鑰材料`signing-key-certificate`，以及步驟 3 的分葉憑證。您也需要提供步驟 1 的匯入金鑰。

6. 使用匯入的金鑰進行密碼編譯操作或後續匯入

   如果匯入的 KeyUsage 是 TR31_K0_KEY_ENCRYPTION_KEY，則此金鑰可用於使用 TR-31 的後續金鑰匯入。如果金鑰類型是任何其他類型（例如 TR31_D0_SYMMETRIC_DATAENCRYPTION_KEY），則金鑰可以直接用於密碼編譯操作。

使用非對稱技術匯入金鑰（RSA取消包裝）

概觀：當 TR-34 不可行時， AWS 付款密碼編譯支援金鑰交換的RSA包裝/取消包裝。與 TR-34 類似，此技術使用RSA非對稱密碼編譯來加密對稱金鑰以進行交換。不過，與 TR-34 不同，此方法沒有由傳送方簽署的承載。此外，由於不包含金鑰區塊，此RSA包裝技術不會在傳輸期間維持金鑰中繼資料的完整性。

> **Note**
>
> RSA 包裝可用於匯入或匯出 TDES和 AES-128 金鑰。

1. 呼叫初始化匯入命令

   在交換金鑰時，呼叫 `get-parameters-for-import` 以 KEY_CRYPTOGRAM.. 的金鑰材料類型初始化匯入程序 WrappingKeyAlgorithm 可以是 RSA_2048TDES。RSA交換 或 AES-128 金鑰時，可以使用 _3072 TDES或 RSA_4096。這API將產生金鑰對，用於金鑰匯入，使用憑證根簽署金鑰，並傳回憑證和憑證根。最後，要匯出的金鑰應該使用此金鑰加密。請注意，這些憑證為短期使用，僅供此用途使用。

```
$ aws payment-cryptography get-parameters-for-import --key-material-type
 KEY_CRYPTOGRAM --wrapping-key-algorithm RSA_4096
```

```
{
    "ImportToken": "import-token-bwxli6ocftypneu5",
    "ParametersValidUntilTimestamp": 1698245002.065,
    "WrappingKeyCertificateChain": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0....",
    "WrappingKeyCertificate": "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0....",
    "WrappingKeyAlgorithm": "RSA_4096"
}
```

2. 在金鑰來源系統上安裝公有憑證

   對於許多 HSMs，您可能需要使用步驟 1 中產生的install/load/trust the public certificate(and/or 根 ），才能使用它匯出金鑰。

3. 從來源系統匯出金鑰

   許多 HSMs和 相關系統支援使用RSA包裝匯出金鑰的功能。您需要將步驟 1 的公有金鑰指定為（加密）憑證（WrappingKeyCertificate）。如果您需要信任鏈，這包含在步驟 #1 WrappingKeyCertificateChain 的回應欄位中。從 匯出金鑰時HSM，您需要將格式指定為 RSA，緩衝模式 = PKCS#1 v2.2 OAEP（使用 SHA 256 或 SHA 512）。

4. 呼叫匯入金鑰

   作為最後一個步驟，您將使用 KeyMaterialType 的 呼叫 importKey API `KeyMaterial`。您需要從步驟 1 匯入金鑰，以及從步驟 3 匯入金鑰材料 `key-material`（已包裝金鑰材料）。您將需要提供金鑰參數（例如金鑰用量），因為RSA包裝不會使用金鑰區塊。

```
$ cat import-key-cryptogram.json
              {
                  "KeyMaterial": {
```

```
                        "KeyCryptogram": {
                            "Exportable": true,
                            "ImportToken": "import-token-bwxli6ocftypneu5",
                            "KeyAttributes": {
                                "KeyAlgorithm": "AES_128",
                                "KeyClass": "SYMMETRIC_KEY",
                                "KeyModesOfUse": {
                                    "Decrypt": true,
                                    "DeriveKey": false,
                                    "Encrypt": true,
                                    "Generate": false,
                                    "NoRestrictions": false,
                                    "Sign": false,
                                    "Unwrap": true,
                                    "Verify": false,
                                    "Wrap": true
                                },
                                "KeyUsage": "TR31_K0_KEY_ENCRYPTION_KEY"
                            },
                            "WrappedKeyCryptogram": "18874746731....",
                            "WrappingSpec": "RSA_OAEP_SHA_256"
                        }
                    }
                }
```

```
$ aws payment-cryptography import-key --cli-input-json file://import-key-
cryptogram.json
```

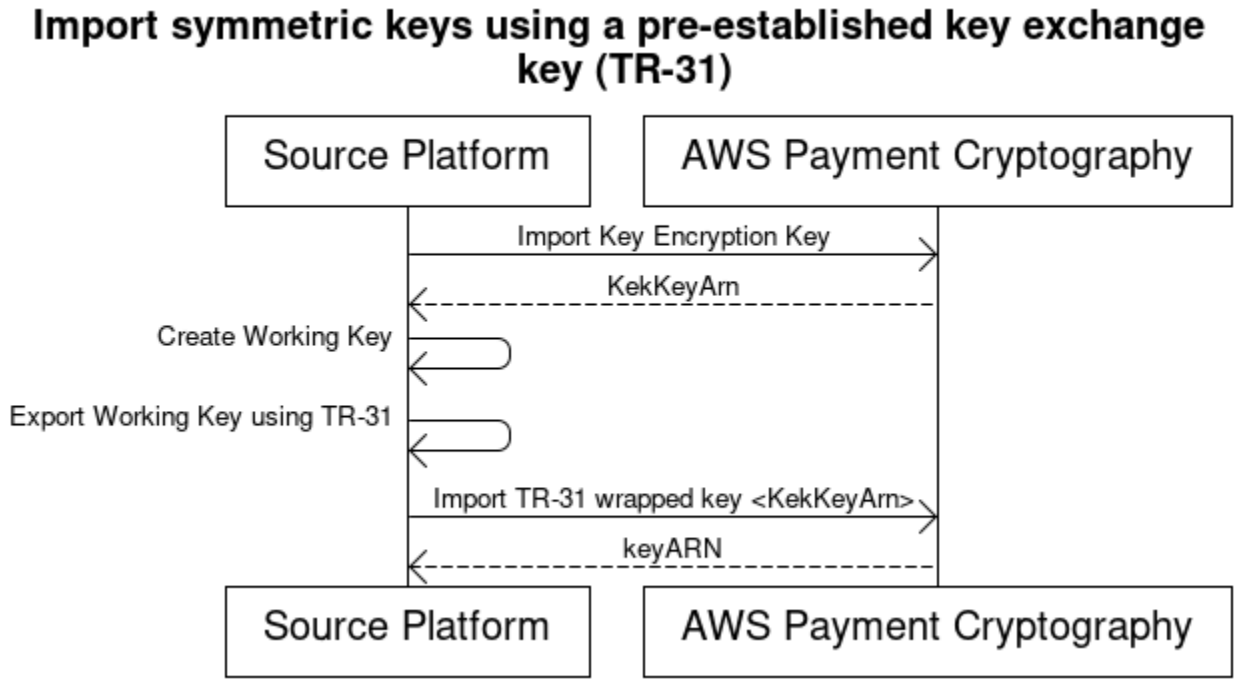
```
{
      "Key": {
          "KeyOrigin": "EXTERNAL",
          "Exportable": true,
          "KeyCheckValue": "DA1ACF",
          "UsageStartTimestamp": 1697643478.92,
          "Enabled": true,
          "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
          "CreateTimestamp": 1697643478.92,
          "KeyState": "CREATE_COMPLETE",
          "KeyAttributes": {
              "KeyAlgorithm": "AES_128",
              "KeyModesOfUse": {
```

```
                "Encrypt": true,
                "Unwrap": true,
                "Verify": false,
                "DeriveKey": false,
                "Decrypt": true,
                "NoRestrictions": false,
                "Sign": false,
                "Wrap": true,
                "Generate": false
            },
            "KeyUsage": "TR31_K0_KEY_ENCRYPTION_KEY",
            "KeyClass": "SYMMETRIC_KEY"
        },
        "KeyCheckValueAlgorithm": "CMAC"
    }
}
```

5. 使用匯入的金鑰進行密碼編譯操作或後續匯入

   如果匯入的 KeyUsage 是 TR31_K0_KEY_ENCRYPTION_KEY，則此金鑰可用於使用 TR-31 的後續金鑰匯入。如果金鑰類型是任何其他類型（例如 TR31_D0_SYMMETRIC_DATAENCRYPTION_KEY），則金鑰可以直接用於密碼編譯操作。

使用預先建立的金鑰交換金鑰（TR-31）匯入對稱金鑰



當合作夥伴交換多個金鑰（或支援金鑰輪換）時，通常會先使用紙本金鑰元件等技術交換初始金鑰加密金鑰（KEK），或使用 TR-34 進行 AWS 付款密碼編譯。

建立 KEK 後，您可以使用此金鑰傳輸後續金鑰（包括其他 KEKs）。 AWS Payment Cryptography 支援使用 ANSI TR-31 進行此類金鑰交換，該金鑰交換受到HSM廠商廣泛使用和支援。

1. 匯入金鑰加密金鑰（KEK）

   假設您已匯入 ，KEK並擁有您可用的金鑰ARN（或 keyAlias）。

2. 在來源平台上建立金鑰

   如果金鑰尚未存在，請在來源平台上建立金鑰。相反地，您可以在 AWS 付款密碼編譯上建立 金鑰，並改用 `export`命令。

3. 從來源平台匯出金鑰

   匯出時，請確定您將匯出格式指定為 TR-31。來源平台也會要求您提供要匯出的金鑰和要使用的金鑰加密金鑰。

4. 匯入 AWS 付款密碼編譯

   呼叫 importKey 命令時， WrappingKeyIdentifier 應該是金鑰加密金鑰ARN的金鑰（或別名） WrappedKeyBlock ，也是來源平台的輸出。

   Example

```
$ aws payment-cryptography import-key \
            --key-material="Tr31KeyBlock={WrappingKeyIdentifier="arn:aws:payment-
cryptography:us-east-2:111122223333:key/ov6icy4ryas4zcza",\

 WrappedKeyBlock="D0112B0AX00E00002E0A3D58252CB67564853373D1EBCC1E23B2ADE7B15E967CC27B85D599
```

```
      {
        "Key": {
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
            "KeyAttributes": {
                "KeyUsage": "TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY",
                "KeyClass": "SYMMETRIC_KEY",
                "KeyAlgorithm": "AES_128",
                "KeyModesOfUse": {
                    "Encrypt": true,
```

```
                    "Decrypt": true,
                    "Wrap": true,
                    "Unwrap": true,
                    "Generate": false,
                    "Sign": false,
                    "Verify": false,
                    "DeriveKey": false,
                    "NoRestrictions": false
                }
            },
            "KeyCheckValue": "0A3674",
            "KeyCheckValueAlgorithm": "CMAC",
            "Enabled": true,
            "Exportable": true,
            "KeyState": "CREATE_COMPLETE",
            "KeyOrigin": "EXTERNAL",
            "CreateTimestamp": "2023-06-02T07:38:14.913000-07:00",
            "UsageStartTimestamp": "2023-06-02T07:38:14.857000-07:00"
        }
    }
```

匯入非對稱（RSA）金鑰

匯入RSA公有金鑰

AWS 付款密碼編譯支援以 X.509 憑證形式匯入公有RSA金鑰。若要匯入憑證，您必須先匯入其根憑證。所有憑證在匯入時都應未過期。憑證應該是 PEM 格式，且應該是 base64 編碼。

1. 匯入根憑證至 AWS 付款密碼編譯

   Example

```
$ aws payment-cryptography import-key \
  --key-material='{"RootCertificatePublicKey":{"KeyAttributes":
{"KeyAlgorithm":"RSA_2048", \
  "KeyClass":"PUBLIC_KEY", "KeyModesOfUse":{"Verify":
 true},"KeyUsage":"TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"}, \

 "PublicKeyCertificate":"LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURKVENDQWcyZ0F3SUJBZ0lCWkR
```

```
  {
```

```
    "Key": {
        "CreateTimestamp": "2023-08-08T18:52:01.023000+00:00",
        "Enabled": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
zabouwe3574jysdl",
        "KeyAttributes": {
            "KeyAlgorithm": "RSA_2048",
            "KeyClass": "PUBLIC_KEY",
            "KeyModesOfUse": {
                "Decrypt": false,
                "DeriveKey": false,
                "Encrypt": false,
                "Generate": false,
                "NoRestrictions": false,
                "Sign": false,
                "Unwrap": false,
                "Verify": true,
                "Wrap": false
            },
            "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"
        },
        "KeyOrigin": "EXTERNAL",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2023-08-08T18:52:01.023000+00:00"
    }
}
```

2. 將公有金鑰憑證匯入 AWS 付款密碼編譯

   您現在可以匯入公有金鑰。匯入公有金鑰有兩種選項。`TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`如果金鑰的用途是驗證簽章（例如，使用 TR-34 匯入時），`TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION`則可以在加密擬用於另一個系統的資料時使用。

   Example

```
$ aws payment-cryptography import-key \
    --key-material='{"TrustedCertificatePublicKey":
{"CertificateAuthorityPublicKeyIdentifier":"arn:aws:payment-cryptography:us-
east-2:111122223333:key/zabouwe3574jysdl", \
    "KeyAttributes":
{"KeyAlgorithm":"RSA_2048","KeyClass":"PUBLIC_KEY","KeyModesOfUse":
{"Verify":true},"KeyUsage":"TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"},\
```

```
    "PublicKeyCertificate":"LS0tLS1CRUdJTiB..."}}'
```

```
{
  "Key": {
      "CreateTimestamp": "2023-08-08T18:55:46.815000+00:00",
      "Enabled": true,
      "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/4kd6xud22e64wcbk",
      "KeyAttributes": {
          "KeyAlgorithm": "RSA_4096",
          "KeyClass": "PUBLIC_KEY",
          "KeyModesOfUse": {
              "Decrypt": false,
              "DeriveKey": false,
              "Encrypt": false,
              "Generate": false,
              "NoRestrictions": false,
              "Sign": false,
              "Unwrap": false,
              "Verify": true,
              "Wrap": false
          },
          "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"
      },
      "KeyOrigin": "EXTERNAL",
      "KeyState": "CREATE_COMPLETE",
      "UsageStartTimestamp": "2023-08-08T18:55:46.815000+00:00"
  }
}
```

匯出金鑰

主題

- 匯出對稱索引鍵
- 匯出非對稱（RSA）金鑰

匯出對稱索引鍵

> **Important**
> 範例可能需要 VAWSCLIV2 的最新版本。開始之前，請確保您已升級至最新版本。

主題

使用非對稱技術匯出金鑰（TR-34）

概觀：TR-34 使用RSA非對稱密碼編譯來加密對稱金鑰以進行交換，並確保資料來源（簽署）。這可確保包裝金鑰的機密性（加密）和完整性（簽章）。匯出時， AWS 付款密碼編譯會成為金鑰分發主機（KDH），而目標系統則會成為金鑰接收裝置（KRD）。

1. 呼叫初始化匯出命令

   呼叫 `get-parameters-for-export` 以初始化匯出程序。API 這會為金鑰匯出產生金鑰對，簽署金鑰並傳回憑證和憑證根。最後，此命令產生的私有金鑰會用來簽署匯出承載。在 TR-34 術語中，這稱為KDH簽署憑證。請注意，這些憑證為短期使用，僅供此用途使用。參數 ParametersValidUntilTimestamp會指定其持續時間。

   NOTE：所有憑證都會以 base64 編碼格式傳回

   Example

```
$ aws payment-cryptography get-parameters-for-export \
                    --signing-key-algorithm RSA_2048 --key-material-type
 TR34_KEY_BLOCK
```

```
{
```

```
                "SigningKeyCertificate":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2RENDQXFTZ0F3SUJBZ0lRZFAzSzNHNEFKT0I4WTNpTmUvYl
                "SigningKeyCertificateChain":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUY0VENDQThtZ0F3SUJBZ0lSQUtlN2piaHFKZjJPd3FGUWI5c3
                "SigningKeyAlgorithm": "RSA_2048",
                "ExportToken": "export-token-au7pvkbsq4mbup6i",
                "ParametersValidUntilTimestamp": "2023-06-13T15:40:24.036000-07:00"
}
```

2. 將 AWS 付款密碼編譯憑證匯入接收系統

   視需要將步驟 1 中提供的憑證鏈匯入接收系統。

3. 產生金鑰對、建立公有憑證並提供 AWS 付款密碼編譯的憑證根目錄

   為了確保傳輸承載的機密性，傳送方會加密（稱為金鑰分發主機或 KDH）。接收方（通常是您HSM或您合作夥伴的 HSM）會想要產生用於此目的的公有金鑰，然後建立可提供給 AWS Payment Cryptography 的公有金鑰憑證（x.509）。 AWS Private CA 是產生憑證的一個選項，但對使用的憑證授權機構沒有限制。

   取得憑證後，您會想要使用 `ImportKey` 命令和 KeyMaterialType 的 和 將根憑證載入 AWS 付款密碼編譯`ROOT_PUBLIC_KEY_CERTIFICATE` KeyUsageType `TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE`。

   此憑證 KeyUsageType 的 是 TR31_S0_ASYMMETRICKEY_FOR__DIGITAL_SIGNATURE，因為它是根金鑰，用於簽署分葉憑證。用於匯入/匯出的分葉憑證不會匯入 AWS 付款密碼編譯，而是內嵌傳遞。

   > **Note**
   > 如果先前已匯入根憑證，則可以略過此步驟。

4. 通話匯出金鑰

   作為最後一個步驟，您將使用 KeyMaterialType 的 呼叫 ExportKey API `TR34_KEY_BLOCK`。`certificate-authority-public-key-identifier` 將是步驟 3 中根 CA 匯入的金鑰ARN，`WrappingKeyCertificate`將是步驟 3 的葉子憑證，`export-key-identifier`也是要匯出的金鑰ARN（或別名）。您也需要提供步驟 1 的匯出金鑰。

使用非對稱技術匯出金鑰（RSA 包裝）

概觀：當 TR-34 不是交易對手可用的選項時， AWS 付款密碼編譯支援金鑰交換的RSA包裝/取消包裝。與 TR-34 類似，此技術使用RSA非對稱密碼編譯來加密對稱金鑰以進行交換。不過，與 TR-34 不同，此方法沒有由傳送方簽署的承載。此外，此RSA包裝技術不包含用於在傳輸期間維護金鑰中繼資料完整性的金鑰區塊。

> **Note**
>
> RSA 包裝可用於匯出 TDES和 AES-128 金鑰。

1. 在接收系統時產生RSA金鑰和憑證

   建立（或識別）將用於接收包裝RSA金鑰的金鑰。 AWS 付款密碼編譯預期金鑰為 X.509 憑證格式。憑證應由匯入（或可匯入）至 AWS Payment Cryptography 的根憑證簽署。

2. 在 AWS 付款密碼編譯上安裝根公有憑證

```
$ aws payment-cryptography  import-key --key-material='{"RootCertificatePublicKey":
{"KeyAttributes":{"KeyAlgorithm":"RSA_4096","KeyClass":"PUBLIC_KEY","KeyModesOfUse":
{"Verify":
 true},"KeyUsage":"TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"},"PublicKeyCertificate":"LS
```

```
    {
        "Key": {
        "CreateTimestamp": "2023-09-14T10:50:32.365000-07:00",
        "Enabled": true,
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
nsq2i3mbg6sn775f",
        "KeyAttributes": {
        "KeyAlgorithm": "RSA_4096",
        "KeyClass": "PUBLIC_KEY",
        "KeyModesOfUse": {
        "Decrypt": false,
        "DeriveKey": false,
        "Encrypt": false,
        "Generate": false,
        "NoRestrictions": false,
        "Sign": false,
        "Unwrap": false,
```

```
        "Verify": true,
        "Wrap": false
        },
        "KeyUsage": "TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE"
        },
        "KeyOrigin": "EXTERNAL",
        "KeyState": "CREATE_COMPLETE",
        "UsageStartTimestamp": "2023-09-14T10:50:32.365000-07:00“
        }
       }
```

3. 呼叫匯出金鑰

接下來，您要指示 AWS Payment Cryptography 使用您的分葉憑證匯出金鑰。您將ARN為先前匯入的根憑證、用於匯出的分葉憑證和要匯出的對稱金鑰指定 。輸出將是對稱金鑰的十六進位編碼二進位封裝（加密）版本。

```
$ cat export-key.json
```

```
  {
      "ExportKeyIdentifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/tqv5yij6wtxx64pi",
      "KeyMaterial": {
          "KeyCryptogram": {
              "CertificateAuthorityPublicKeyIdentifier": "arn:aws:payment-
cryptography:us-east-2:111122223333:key/zabouwe3574jysdl",
              "WrappingKeyCertificate": "LS0tLS1CRUdJTiBD...",
              "WrappingSpec": "RSA_OAEP_SHA_256"
          }
      }
  }
```

```
$ aws payment-cryptography export-key --cli-input-json file://export-key.json
```

```
{
      "WrappedKey": {
          "KeyMaterial":
 "18874746731E9E1C4562E4116D1C2477063FCB08454D757D81854AEAEE0A52B1F9D303FA29C02DC82AE7785353
          "WrappedKeyMaterialFormat": "KEY_CRYPTOGRAM"
```

```
        }
    }
```

4. 匯入金鑰至接收系統

   許多 HSMs和 相關系統支援使用 RSA unwrap 匯入金鑰的功能（包括 AWS 付款密碼編譯）。若要這麼做，請將步驟 1 的公有金鑰指定為（加密）憑證，且格式應指定為 RSA，Padding Mode = PKCS#1 v2.2 OAEP（含 SHA 256）。確切的術語可能因 而異HSM。

   > **Note**
   >
   > AWS 付款密碼編譯會在 中輸出封裝的金鑰hexBinary。如果您的系統需要不同的二進位表示法，例如 base64，您可能需要在匯入之前轉換格式。

使用預先建立的金鑰交換金鑰（TR-31）匯出對稱金鑰

當合作夥伴交換多個金鑰（或支援金鑰輪換）時，通常會先使用紙本金鑰元件等技術交換初始金鑰加密金鑰（KEK），或使用 TR-34 進行 AWS 付款密碼編譯。建立 KEK 後，您可以使用此金鑰傳輸後續金鑰（包括其他 KEK）。 AWS Payment Cryptography 支援使用 ANSI TR-31 的此類金鑰交換，其廣泛使用且受到HSM廠商廣泛支援。

1. Exchange 金鑰加密金鑰（KEK）

   假設您已交換 ，KEK並擁有您可用的金鑰ARN（或 keyAlias）。

2. 在 AWS 付款密碼編譯上建立金鑰

   如果金鑰尚未存在，請建立金鑰。相反地，您可以在其他系統上建立金鑰，並改用匯入命令。

3. 從 AWS 付款密碼編譯匯出金鑰

   匯出時，格式將為 TR-31。呼叫 時API，您將指定要匯出的金鑰和要使用的包裝金鑰。

```
$ aws payment-cryptography export-key --key-material='{"Tr31KeyBlock":
 {"WrappingKeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza"}}' --export-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp
```

```
{
                "WrappedKey": {
                    "KeyCheckValue": "73C263",
                    "KeyCheckValueAlgorithm": "ANSI_X9_24",
```

```
                    "KeyMaterial":
"D0144K0AB00E0000A24D3ACF3005F30A6E31D533E07F2E1B17A2A003B338B1E79E5B3AD4FBF7850FACF9A37844
                    "WrappedKeyMaterialFormat": "TR31_KEY_BLOCK"
                }
        }
```

4. 匯入至您的系統

   您或您的合作夥伴將在系統上使用匯入金鑰實作來匯入金鑰。

匯出DUKPT初始金鑰（IPEK/IK）

使用 時DUKPT，可能會為終端機機群產生單一 Base Derivation Key（BDK）。不過，終端永遠無法存取該原始伺服器，BDK但每個裝置都會注入一個唯一的初始終端金鑰，稱為 IPEK或 初始金鑰（IK）。每個 IPEK 都是衍生自 的金鑰，BDK旨在為每個終端機唯一，但衍生自原始 BDK。此計算的衍生資料稱為金鑰序號（KSN）。根據 X9.24，TDES對於 10 位元組，KSN通常包含 24 位元的金鑰集 ID、19 位元的終端機 ID 和 21 位元的交易計數器。對於 AES，12 位元組KSN通常包含 32 位元的 BDK ID、32 位元的衍生識別符（ID），以及 32 位元的交易計數器。

AWS 付款密碼編譯提供產生和匯出這些初始金鑰的機制。產生後，即可使用 TR-31, TR-34 和RSA包裝方法匯出這些金鑰。IPEK 金鑰不會持續存在，也無法用於 AWS 付款密碼編譯的後續操作

AWS 付款密碼編譯不會強制執行 的前兩個部分之間的分割KSN。如果您想要將衍生識別符與 一起存放BDK，您可以將 AWS 標籤功能用於此目的。

> **Note**
>
> KSN（ 的 32 位元DUKPT）AES 的計數器部分不會用於 IPEK/IK 衍生。因此，輸入 12345678901234560001 和 12345678901234569999 將輸出相同的 IPEK。

```
$ aws payment-cryptography export-key --key-material='{"Tr31KeyBlock":
 {"WrappingKeyIdentifier": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza"}}' --export-key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi --export-attributes
 'ExportDukptInitialKey={KeySerialNumber=12345678901234560001}'
```

```
{
    "WrappedKey": {
        "KeyCheckValue": "73C263",
```

```
        "KeyCheckValueAlgorithm": "ANSI_X9_24",
        "KeyMaterial":
 "B0096B1TX00S000038A8A06588B9011F0D5EEF1CCAECFA6962647A89195B7A98BDA65DDE7C57FEA507559AF2A5D60
        "WrappedKeyMaterialFormat": "TR31_KEY_BLOCK"
    }
}
```

匯出時指定金鑰區塊標頭

AWS 付款密碼編譯支援在以 ASC TR-31 或 TR-34 格式匯出時修改或附加金鑰區塊資訊。下表說明 TR-31 金鑰區塊格式，以及匯出時可以修改哪些元素。

金鑰區塊屬性	用途	匯出時修改或附加？	備註
版本 ID	代表用來保護基礎金鑰材料的方法。此標準定義了版本 A 和版本 C （金鑰變體）、版本 B （使用衍生TDES） 和版本 D （使用 衍生金鑰 AES）。	無法修改	AWS 包裝金鑰為 時，付款密碼編譯將使用版本 B，包裝金鑰為時，將使用TDES版本 DAES。版本 A 和版本 C 已棄用，且僅支援匯入操作。
金鑰區塊長度	代表剩餘訊息的長度	無法修改	此值由 服務自動計算。請注意，服務可能會根據規格的需求使用金鑰填充，因此在解密承載之前，長度可能不會正確顯示。
金鑰使用方式	表示金鑰的允許用途，例如 C0 （卡片驗證） 或 B0 （基本衍生金鑰）	無法修改	
演算法	代表基礎金鑰的演算法。此標準支援多種金鑰類型，例如	無法修改	這會從 AWS Payment Cryptography as-is 匯出。

金鑰區塊屬性	用途	匯出時修改或附加？	備註
	T （TDES）、A（AES） 和 E（ ）ECC。 AWS 付款密碼編譯目前支援 T、H 和 A 的值。		
金鑰使用方式	表示可用於 的操作類型。範例包括產生和驗證 （C） 和 Encrypt/Decrypt/Wrap/Unwrap（B）	修改*	
金鑰版本	如果金鑰已取代/輪換，則表示金鑰的版本編號。這是數值，如果未指定，則預設為 00。	附加	
金鑰可匯出性	表示是否可以匯出金鑰。N 表示無可匯出性，E 表示根據 X9.24（金鑰區塊） 匯出，S 表示以金鑰區塊或非金鑰區塊格式匯出。	修改*	
選用金鑰區塊	附加	選用的金鑰區塊是一系列名稱/值對，可加密綁定至金鑰。常見的範例是DUKPT金鑰的 KeySetID。實際格式包括選用區塊的數量、每個區塊的長度和填充區塊 （PB），但 AWS 付款密碼編譯會根據名稱/值對輸入自動計算這些區塊。	

*修改時，新值必須比 AWS Payment Cryptography 中的目前值更嚴格。例如，如果目前的金鑰使用模式是 Generate=True，Verify=True，則匯出時，您可以將其變更為 Generate=True，Verify=False。同樣地，如果金鑰已設定為無法匯出，則您無法將其變更為可匯出。

匯出金鑰時， AWS 付款密碼編譯會自動為匯出的金鑰套用目前的值，而不進行修改。不過，在某些情況下，您可能想要在傳送至接收系統之前修改或附加這些值，即使 AWS 付款密碼編譯中允許這樣做。其中一個範例是，將金鑰匯出至付款終端機時，通常會限制其可匯出性為 ，`Not Exportable`因為終端機通常只用來匯入金鑰，因此不應後續匯出金鑰。

另一個範例是當您想要將相關聯的金鑰中繼資料傳遞至接收系統時。在 TR-31 之前，常見做法是建立自訂承載以傳送這類資訊，但使用 TR-31，您可以截斷方式將其繫結至特定格式的金鑰。可以使用 `KeyVersion` 欄位設定金鑰版本。TR-31/X9.143 會指定特定的常見標頭，但只要符合 AWS 付款密碼編譯參數且接收系統能夠接受，就不會限制使用其他標頭。如需有關在匯出期間指定金鑰區塊的詳細資訊，請參閱 API指南中的金鑰區塊標頭。

在此範例中，我們正在匯出BDK金鑰 （ 例如 ）KIF。我們會將金鑰版本指定為 02， KeyExportability 將 設定為 NON_EXPORTABLE，並為 KeySetID 提供 00ABCDEFAB 的選用值，這表示它是TDES 金鑰 （ 00 ），而初始金鑰是 ABCDEFABCD。由於未指定金鑰使用模式，此金鑰會繼承 arn：aws：payment-cryptography：us-east-2：111122223333：key/5rplquuwozodpwsp 的使用模式，其為 `DeriveKey = true`

> **Note**
>
> 即使在此範例中將可匯出性設定為不可匯出， KIF仍然可以衍生金鑰，例如 中使用的 IPEK/IKDUKPT，而且這些衍生金鑰可以匯出，以便在裝置上安裝 。該標準特別允許這樣做。

```
$ aws payment-cryptography --key-material='{"Tr31KeyBlock":
 {"WrappingKeyIdentifier":"arn:aws:payment-cryptography:us-
east-2:111122223333:key/ov6icy4ryas4zcza","KeyBlockHeaders":{"KeyModesOfUse":
{"Derive":true},"KeyExportability":"NON_EXPORTABLE","KeyVersion":"02","OptionalBlocks":
{"BI":"00ABCDEFABCD"}}}} --export-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp
```

```
{
        "WrappedKey": {
            "WrappedKeyMaterialFormat": "TR31_KEY_BLOCK",
            "KeyMaterial":
 "D0128B0TX02N0100BI1000ABCDEFABCD1A2C0EADE244321640E94FE3A3C9D33800D47CE64238D9327DDBFE25B9023
            "KeyCheckValue": "A4C9B3",
```

```
            "KeyCheckValueAlgorithm": "ANSI_X9_24"
        }
  }
```

匯出非對稱（RSA）金鑰

呼叫 `get-public-key-certificate`以憑證形式匯出公有金鑰。這API將匯出以 base64 格式編碼的憑證及其根憑證。

NOTE：API這不具有同位 - 後續呼叫可能會導致不同的憑證，即使基礎金鑰相同。

Example

```
$ aws payment-cryptography get-public-key-certificate \
         –key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/5dza7xqd6soanjtb
```

```
{
        "KeyCertificate": "LS0tLS1CRUdJTi...",
        "KeyCertificateChain": "LS0tLS1CRUdJT...“
}
```

使用別名

別名是 AWS 付款密碼編譯金鑰的易記名稱。例如，別名可讓您將金鑰稱為 `alias/test-key`，而不是 `arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h`。

您可以使用別名，在大多數金鑰管理（控制平面）操作和密碼編譯（資料平面）操作 中識別金鑰。

您也可以允許和拒絕根據其別名存取 AWS Payment Cryptography 金鑰，而無需編輯政策或管理授予。此功能是 服務對屬性型存取控制（ ）支援的一部分ABAC。

別名的大部分功能來自您隨時變更與別名相關聯的金鑰。別名可以讓您的程式碼更容易撰寫和維護。例如，假設您使用別名來參考特定的 AWS 付款密碼編譯金鑰，而且您想要變更 AWS 付款密碼編譯金鑰。在這種情況下，只要將別名與不同的金鑰建立關聯即可。您不需要變更程式碼或應用程式組態。

別名也可以更容易在不同的 AWS 區域中重複使用相同的程式碼。在多個區域中建立具有相同名稱的別名，並將每個別名與其區域中的 AWS 付款密碼編譯金鑰建立關聯。當程式碼在每個區域中執行時，別名會參考該區域中相關聯的 AWS 付款密碼編譯金鑰。

您可以使用 為 AWS 付款密碼編譯金鑰建立別名`CreateAlias`API。

AWS 付款密碼編譯API提供每個帳戶和區域中別名的完整控制。API 包含建立別名 （CreateAlias）、檢視別名名稱和連結金鑰ARN （list-aliases ） 的操作、變更與別名 （update-alias ） 相關聯的 AWS 付款密碼編譯金鑰，以及刪除別名 （delete-alias ）。

主題

關於別名

了解別名如何在 AWS 付款密碼編譯中運作。

別名是獨立 AWS 資源

別名不是 AWS Payment Cryptography 金鑰的屬性。您在別名上採取的動作不會影響其相關聯的金鑰。您可以為 AWS 付款密碼編譯金鑰建立別名，然後更新別名，使其與不同的 AWS 付款密碼編譯金鑰相關聯。您甚至可以刪除別名，而不會影響相關聯的 AWS Payment Cryptography 金鑰。如果您刪除 AWS 付款密碼編譯金鑰，與該金鑰相關聯的所有別名都會變成未指派。

如果您將別名指定為IAM政策中的資源，則政策會參考別名，而不是相關聯的 AWS 付款密碼編譯金鑰。

每個別名都有一個易記的名稱

建立別名時，您可以指定 字首的別名名稱`alias/`。例如 `alias/test_1234`

每個別名一次與一個 AWS 付款密碼編譯金鑰相關聯

別名及其 AWS 付款密碼編譯金鑰必須位於相同的帳戶和區域中。

AWS 付款密碼編譯金鑰可以同時與多個別名建立關聯，但每個別名只能對應至單一金鑰

例如，此`list-aliases`輸出顯示別名僅與一個由 `KeyArn` 屬性表示的目標 AWS Payment Cryptography `alias/sampleAlias1` 金鑰相關聯。

```
$ aws payment-cryptography list-aliases
```

```
  {
    "Aliases": [
        {
            "AliasName": "alias/sampleAlias1",
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
        }
    ]
}
```

多個別名可以與相同的 AWS 付款密碼編譯金鑰相關聯

例如，您可以將 `alias/sampleAlias2` `alias/sampleAlias1;`和 別名與相同的金鑰建立關聯。

```
$ aws payment-cryptography list-aliases
```

```
{
        "Aliases": [
            {
                "AliasName": "alias/sampleAlias1",
                "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
            },
            {
                "AliasName": "alias/sampleAlias2",
                "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
            }
        ]
    }
```

指定帳戶和區域的別名必須是唯一的

例如，您在每個帳戶和區域只能有一個 `alias/sampleAlias1` 別名。別名區分大小寫，但我們建議不要使用大小寫不同的別名，因為它們可能容易出錯。您無法變更別名名稱。但是，您可以刪除別名，並使用所需名稱建立新別名。

但是，您可以在不同區域中使用相同的名稱建立別名。

例如，您可以在alias/sampleAlias2美國東部（維吉尼亞北部）有別名，在美國alias/sampleAlias2西部（奧勒岡）有別名。每個別名都會與其區域中的 AWS 付款密碼編譯金鑰相關聯。如果您的程式碼引用了類似 alias/finance-key 的別名名稱，則可以在多個區域中執行。在每個區域中，它使用不同的別名/sampleAlias2。如需詳細資訊，請參閱 在應用程式中使用別名。

您可以變更與別名相關聯的 AWS 付款密碼編譯金鑰

您可以使用 UpdateAlias操作，將別名與不同的 AWS 付款密碼編譯金鑰建立關聯。例如，如果別名與 arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h AWS Payment Cryptography alias/sampleAlias2 金鑰相關聯，您可以更新它，使其與arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi金鑰相關聯。

> **Warning**
>
> AWS 付款密碼編譯不會驗證舊金鑰和新金鑰具有所有相同的屬性，例如金鑰用量。使用不同的金鑰類型更新可能會導致應用程式發生問題。

某些金鑰沒有別名

別名是選用功能，除非您選擇以這種方式操作環境，否則並非所有金鑰都會有別名。金鑰可以使用 create-alias命令與別名建立關聯。此外，您可以使用 update-alias 操作來變更與別名相關聯的 AWS 付款密碼編譯金鑰，並使用 delete-alias 操作來刪除別名。因此，有些 AWS 付款密碼編譯金鑰可能具有多個別名，有些則可能沒有。

將索引鍵映射至別名

您可以使用 create-alias命令將金鑰（由 表示ARN）對應至一或多個別名。此命令不是 idempotent - 若要更新別名，請使用 update-alias 命令。

```
$ aws payment-cryptography create-alias --alias-name alias/sampleAlias1 \
             --key-arn arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h
```

```
{
    "Alias": {
        "AliasName": "alias/alias/sampleAlias1",
```

```
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
    }
}
```

在應用程式中使用別名

您可以使用別名來代表應用程式程式碼中的 AWS 付款密碼編譯金鑰。 AWS 付款密碼編譯資料操作中的 `key-identifier` 參數，以及 List Keys 等其他操作接受別名名稱或別名 ARN。

```
$ aws payment-cryptography-data generate-card-validation-data --key-identifier alias/
BIN_123456_CVK --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue2={CardExpiryDate=0123}
```

使用別名 時ARN，請記住，對應至 AWS Payment Cryptography 金鑰的別名是在擁有 AWS Payment Cryptography 金鑰的帳戶中定義，而且在每個區域中可能會有所不同。

其中一個最強大的別名用途是在多個 AWS 區域中執行之應用程式中使用。

您可以在每個區域中建立不同的應用程式版本，或使用字典、組態或切換陳述式來為每個區域選取正確的 AWS 付款密碼編譯金鑰。但在每個區域中建立具有相同別名名稱的別名可能比較容易。請記住，別名名稱區分大小寫。

相關 APIs

Tags (標籤)

標籤是做為中繼資料的金鑰和值對，用於組織您的 AWS 付款密碼編譯金鑰。它們可用來靈活識別金鑰，或將一或多個金鑰分組在一起。

取得金鑰

AWS 付款密碼編譯金鑰代表單一單位的密碼編譯材料，只能用於此服務的密碼編譯操作。以 GetKeys API KeyIdentifier 作為輸入，並傳回金鑰的不可變和不可變屬性，但不包含任何密碼編譯材料。

Example

```
$ aws payment-cryptography get-key --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/kwapwa6qaifllw2h
```

```
{
  "Key": {
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
      "KeyAttributes": {
          "KeyUsage": "TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY",
          "KeyClass": "SYMMETRIC_KEY",
          "KeyAlgorithm": "AES_128",
          "KeyModesOfUse": {
              "Encrypt": true,
              "Decrypt": true,
              "Wrap": true,
              "Unwrap": true,
              "Generate": false,
              "Sign": false,
              "Verify": false,
              "DeriveKey": false,
              "NoRestrictions": false
          }
      },
      "KeyCheckValue": "0A3674",
      "KeyCheckValueAlgorithm": "CMAC",
      "Enabled": true,
      "Exportable": true,
      "KeState": "CREATE_COMPLETE",
      "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
      "CreateTimestamp": "2023-06-02T07:38:14.913000-07:00",
      "UsageStartTimestamp": "2023-06-02T07:38:14.857000-07:00"
  }
}
```

取得與金鑰對相關聯的公有金鑰/憑證

Get Public Key/Certificate 會傳回 指示的公有金鑰`KeyArn`。這可以是在 AWS 付款密碼編譯上產生的金鑰對公有金鑰部分，也可以是先前匯入的公有金鑰。最常見的使用案例是將公有金鑰提供給會加密資

料的外部服務。然後，該資料可以傳遞到利用 AWS Payment Cryptography 的應用程式，並且可以使用在 AWS Payment Cryptography 中保護的私有金鑰解密資料。

服務會將公有金鑰傳回為公有憑證。API 結果包含 CA 和公有金鑰憑證。這兩個資料元素都是 base64 編碼。

> **Note**
>
> 傳回的公有憑證旨在短暫使用，且並非意在具有意向性。即使公有金鑰本身不變，每次API通話時您都可能會收到不同的憑證。

Example

```
$ aws payment-cryptography get-public-key-certificate  --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/nsq2i3mbg6sn775f
```


```
{
  "KeyCertificate":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUV2VENDQXFXZ0F3SUJBZ0lSQUo1OWd2VkpDd3dlYldMNldYZEpYY
  "KeyCertificateChain":
 "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUY0VENDQThtZ0F3SUJBZ0lSQUtlN2piaHFKZjJPd3FGUWI5c3VuC
}
```


標記金鑰

在 AWS 付款密碼編譯中，您可以在建立金鑰 時將標籤新增至 AWS 付款密碼編譯金鑰，並標記或取消標記現有金鑰，除非它們處於待刪除狀態。標籤是選用的，但它們可以非常有用。

如需標籤的一般資訊，包括最佳實務、標記策略，以及標籤的格式和語法，請參閱 中的標記 AWS 資源Amazon Web Services 一般參考。

主題

- 關於 AWS 付款密碼編譯中的標籤
- 在主控台中檢視金鑰標籤
- 使用 API 操作管理金鑰標籤
- 控制對標籤的存取
- 使用標籤控制對金鑰的存取

關於 AWS 付款密碼編譯中的標籤

標籤是選用的中繼資料標籤，您可以指派（或 AWS 指派）至 AWS 資源。每個標籤皆包含標籤索引鍵和標籤值，它們都是區分大小寫的字串。此標籤值可以是空 (null) 字串。資源上的每個標籤都必須有不同的標籤金鑰，但您可以將相同的標籤新增至多個 AWS 資源。每個資源最多可以有 50 個使用者建立的標籤。

請勿在標籤金鑰或標籤值包含機密或敏感資訊。許多 都可以存取標籤 AWS 服務，包括帳單。

在 AWS 付款密碼編譯中，您可以在建立金鑰 時將標籤新增至金鑰，並標記或取消標記現有金鑰，除非它們處於待刪除狀態。您無法標記別名。標籤是選用的，但它們可以非常有用。

例如，您可以將`"Project"="Alpha"`標籤新增至您用於 Alpha 專案的所有 AWS Payment Cryptography 金鑰和 Amazon S3 儲存貯體。另一個範例是將`"BIN"="20130622"`標籤新增至與特定銀行識別號碼（）相關聯的所有金鑰BIN。

```
    [
    {
      "Key": "Project",
      "Value": "Alpha"
    },
    {
      "Key": "BIN",
      "Value": "20130622"
    }
  ]
```

如需標籤的一般資訊，包括格式和語法，請參閱 中的標記 AWS 資源Amazon Web Services 一般參考。

標籤可協助您執行以下操作：

- 識別和組織您的 AWS 資源。許多 AWS 服務支援標記，因此您可以將相同的標籤指派給來自不同 服務的資源，以指出資源相關。例如，您可以將相同的標籤指派給 AWS 付款密碼編譯金鑰和 Amazon Elastic Block Store （AmazonEBS） 磁碟區或 AWS Secrets Manager 秘密。您也可以使用標籤來識別自動化的金鑰。
- 追蹤您的 AWS 成本。當您將標籤新增至 AWS 資源時， AWS 會產生成本分配報告，其中包含使用量和標籤彙總的成本。您可以使用此功能來追蹤專案、應用程式或成本中心的 AWS 付款密碼編譯成本。

  如需有關使用成本配置標籤的詳細資訊，請參閱《AWS Billing 使用者指南》中的使用成本分配標籤。如需標籤鍵和標籤值規則的相關資訊，請參閱《AWS Billing 使用者指南》中的使用者定義的標籤限制。
- 控制對 AWS 資源的存取。允許和拒絕根據其標籤存取金鑰是屬性型存取控制 （ ） 的 AWS 付款密碼編譯支援的一部分ABAC。如需根據其標籤控制 AWS Payment Cryptography 存取的資訊，請參閱 以 AWS Payment Cryptography 標籤為基礎的授權。如需使用標籤控制 AWS 資源存取的一般資訊，請參閱 IAM 使用者指南 中的使用資源標籤控制 AWS 資源存取。

AWS 使用 TagResource、 或 ListTagsForResource 操作時 UntagResource，付款密碼編譯會將項目寫入 AWS CloudTrail 日誌。

在主控台中檢視金鑰標籤

若要在主控台中檢視標籤，您需要從包含 金鑰IAM的政策中標記金鑰的許可。除了在主控台中檢視金鑰的許可之外，您還需要這些許可。

使用 API 操作管理金鑰標籤

您可以使用AWS 付款密碼編譯API來新增、刪除和列出您管理之金鑰的標籤。以下範例使用 AWS Command Line Interface (AWS CLI)，但您可以使用任何支援的程式設計語言。您無法標記 AWS 受管金鑰。

若要新增、編輯、檢視和刪除金鑰的標籤，您必須具有所需的許可。如需詳細資訊，請參閱 控制對標籤的存取。

主題

- CreateKey：將標籤新增至新金鑰
- TagResource：新增或變更金鑰的標籤
- ListResourceTags：取得金鑰的標籤

- UntagResource：從金鑰刪除標籤

CreateKey：將標籤新增至新金鑰

您可以在建立金鑰時新增標籤。若要指定標籤，請使用 CreateKey操作的 `Tags` 參數。

若要在建立金鑰時新增標籤，呼叫者必須在IAM政策中擁有`payment-cryptography:TagResource`許可。許可至少必須涵蓋帳戶和區域中的所有金鑰。如需詳細資訊，請參閱 控制對標籤的存取。

`CreateKey` 的 `Tags` 參數值是區分大小寫的標籤鍵和標籤值對的集合。金鑰上的每個標籤都必須具有不同的標籤名稱。標籤值可以為 null 或空字串。

例如，下列 AWS CLI 命令會建立具有`Project:Alpha`標籤的對稱加密金鑰。指定多個索引鍵/值組時，請使用空格來分隔每一組。

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY, \
       KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY, \
       KeyModesOfUse='{Generate=true,Verify=true}' \
       --tags '[{"Key":"Project","Value":"Alpha"},{"Key":"BIN","Value":"123456"}]'
```

當此命令成功時，它會傳回包含新金鑰相關資訊的`Key`物件。但是，`Key` 不包含標籤。若要取得標籤，請使用 ListResourceTags操作。

TagResource：新增或變更金鑰的標籤

TagResource 操作會將一或多個標籤新增至金鑰。您無法使用此操作新增或編輯不同 AWS 帳戶中的標籤。

若要新增標籤，請指定新標籤索引鍵和標籤值。若要編輯標籤，請指定現有標籤索引鍵和新標籤值。金鑰上的每個標籤都必須具有不同的標籤金鑰。標籤值可以為 null 或空字串。

例如，下列命令會將 **`UseCase`**和 **`BIN`** 標籤新增至範例金鑰。

```
$ aws payment-cryptography tag-resource --resource-arn arn:aws:payment-
cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h --tags
 '[{"Key":"UseCase","Value":"Acquiring"},{"Key":"BIN","Value":"123456"}]'
```

當此命令成功時，不會傳回任何輸出。若要檢視金鑰上的標籤，請使用 ListResourceTags操作。

您也可以使用 TagResource 變更現有標籤的標籤值。若要取代標籤值，請使用不同的值來指定相同的標籤索引鍵。未列於修改命令中的標籤不會變更或移除。

例如，這個命令會將 `Project` 標籤的值從 `Alpha` 變更為 `Noe`。

命令將傳回 http/200，不含任何內容。若要查看您的變更，請使用 `ListTagsForResource`

```
$ aws payment-cryptography tag-resource --resource-arn arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h \
        --tags '[{"Key":"Project","Value":"Noe"}]'
```

ListResourceTags：取得金鑰的標籤

ListResourceTags 操作會取得金鑰的標籤。`ResourceArn`（keyArn 或 keyAlias）參數為必填。您無法使用此操作來檢視不同 中金鑰上的標籤 AWS 帳戶。

例如，下列命令會取得範例金鑰的標籤。

```
$ aws payment-cryptography list-tags-for-resource --resource-arn arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h

  {
    "Tags": [
        {
            "Key": "BIN",
            "Value": "20151120"
        },
        {
            "Key": "Project",
            "Value": "Production"
        }
    ]
}
```

UntagResource：從金鑰刪除標籤

UntagResource 操作會從金鑰刪除標籤。若要識別要刪除的標籤，請指定標籤索引鍵。您無法使用此操作從不同的 索引鍵刪除標籤 AWS 帳戶。

成功時，`UntagResource` 操作不會傳回任何輸出。此外，如果在金鑰上找不到指定的標籤金鑰，則不會擲回例外狀況或傳回回應。若要確認操作是否正常運作，請使用 ListResourceTags操作。

例如，此命令會從指定的金鑰刪除**Purpose**標籤及其值。

```
$ aws payment-cryptography untag-resource \
        --resource-arn arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h --tag-keys Project
```

控制對標籤的存取

若要使用 新增、檢視和刪除標籤API，主體需要在IAM政策中標記許可。

您也可以使用標籤的 AWS 全域條件索引鍵來限制這些許可。在 AWS 付款密碼編譯中，這些條件可以控制對標記操作的存取，例如 TagResource和 UntagResource。

如需政策和詳細資訊，請參閱 IAM 使用者指南 中的根據標籤金鑰控制存取。

建立和管理標籤的許可如下所示。

payment-cryptography：TagResource

允許主體新增或編輯標籤。若要在建立金鑰時新增標籤，主體必須具有不限於特定金鑰IAM的政策許可。

payment-cryptography：ListTagsForResource

允許主體檢視金鑰上的標籤。

payment-cryptography：UntagResource

允許主體從金鑰刪除標籤。

標記政策中的許可

您可以在金鑰政策中提供標記許可IAM。例如，下列範例金鑰政策會為選取的使用者提供金鑰的標記許可。它為所有可以擔任範例管理員或開發人員角色的使用者提供檢視標籤的許可。

```
{
  "Version": "2012-10-17",
  "Id": "example-key-policy",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
```

```
      "Action": "payment-cryptography:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow all tagging permissions",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:user/LeadAdmin",
        "arn:aws:iam::111122223333:user/SupportLead"
      ]},
      "Action": [
          "payment-cryptography:TagResource",
          "payment-cryptography:ListTagsForResource",
          "payment-cryptography:UntagResource"
      ],
      "Resource": "*"
    },
    {
      "Sid": "Allow roles to view tags",
      "Effect": "Allow",
      "Principal": {"AWS": [
        "arn:aws:iam::111122223333:role/Administrator",
        "arn:aws:iam::111122223333:role/Developer"
      ]},
      "Action": "payment-cryptography:ListResourceTags",
      "Resource": "*"
    }
  ]
}
```

若要在多個金鑰上授予主體標記許可，您可以使用 IAM政策。為了使此政策有效，每個金鑰的金鑰政策必須允許帳戶使用IAM政策來控制對金鑰的存取。

例如，下列IAM政策允許主體建立金鑰。它還允許它們在指定帳戶中的所有金鑰上建立和管理標籤。此組合可讓主體在建立金鑰時，使用 CreateKey 操作的標籤參數將標籤新增至金鑰。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKeys",
      "Effect": "Allow",
      "Action": "payment-cryptography:CreateKey",
```

```
      "Resource": "*"
    },
    {
      "Sid": "IAMPolicyTags",
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:TagResource",
        "payment-cryptography:UntagResource",
        "payment-cryptography:ListTagsForResource"
      ],
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*"
    }
  ]
}
```

限制標籤許可

您可以使用政策條件來限制標記許可。下列政策條件可套用至 `payment-cryptography:TagResource` 和 `payment-cryptography:UntagResource` 許可。例如，您可以使用 `aws:RequestTag/`*`tag-key`* 條件，允許主體僅新增特定標籤，或防止主體新增具有特定標籤索引鍵的標籤。

- aws：RequestTag
- aws：ResourceTag/tag-key （IAM 僅限 政策）
- aws：TagKeys

作為使用標籤來控制金鑰存取的最佳實務，請使用 `aws:RequestTag/`*`tag-key`*或 `aws:TagKeys`條件金鑰來決定允許哪些標籤 （或標籤金鑰）。

例如，下列IAM政策類似於上一個政策。不過，此政策允許主體建立標籤 (`TagResource`) 並僅為具有 `Project` 標籤索引鍵的標籤刪除標籤 `UntagResource`。

由於 `TagResource`和 `UntagResource`請求可以包含多個標籤，因此您必須指定具有 aws：TagKeys 條件的 `ForAllValues`或 `ForAnyValue` 設定運算子。`ForAnyValue` 運算子會要求請求中的至少一個標籤索引鍵與政策中的標籤索引鍵相符。`ForAllValues` 運算子會要求請求中的所有標籤索引鍵與政策中的其中一個標籤索引鍵相符。`true` 如果請求中沒有標籤，但 UntagResource沒有指定標籤時 TagResource 失敗，`ForAllValues`則運算子也會傳回 。如需集運算子的詳細資訊，請參閱 IAM 使用者指南 中的使用多個金鑰和值。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyCreateKey",
      "Effect": "Allow",
      "Action": "payment-cryptography:CreateKey",
      "Resource": "*"
    },
    {
      "Sid": "IAMPolicyViewAllTags",
      "Effect": "Allow",
      "Action": "payment-cryptography:ListResourceTags",
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*"
    },
    {
      "Sid": "IAMPolicyManageTags",
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:TagResource",
        "payment-cryptography:UntagResource"
      ],
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*",
      "Condition": {
          "ForAllValues:StringEquals": {"aws:TagKeys": "Project"}
      }
    }
  ]
}
```

使用標籤控制對金鑰的存取

您可以根據金鑰上的標籤來控制 AWS 對付款密碼編譯的存取。例如，您可以撰寫IAM政策，允許主體僅啟用和停用具有特定標籤的金鑰。或者，您可以使用 IAM政策來防止主體在密碼編譯操作中使用金鑰，除非金鑰具有特定標籤。

此功能是屬性型存取控制（ ）的 AWS 付款密碼編譯支援的一部分ABAC。如需使用標籤控制資源存取 AWS 的相關資訊，請參閱 使用者指南 中的什麼ABAC是 AWS？ 和使用資源標籤控制 AWS 資源存取。 IAM

> **Note**

AWS Payment Cryptography 支援 aws：ResourceTag/tag-key 全域條件內容金鑰，可讓您根據金鑰上的標籤控制對金鑰的存取。由於多個金鑰可以具有相同的標籤，此功能可讓您將許可套用至一組選取的金鑰。您也可以透過變更金鑰標籤，輕鬆變更 集中的金鑰。

在 AWS 付款密碼編譯中，僅IAM政策支援`aws:ResourceTag/`*`tag-key`*條件金鑰。金鑰政策不支援此功能，僅適用於一個金鑰，或不使用特定金鑰的操作，例如 ListKeys或 ListAliases操作。

使用標籤控制存取可提供一種簡單、可擴展且靈活的方式來管理許可。但是，如果沒有正確設計和管理，可能會不慎允許或拒絕存取您的金鑰。如果您使用標籤來控制存取，請考慮下列實務。

- 使用標籤來強化最低權限存取的最佳實務。僅提供IAM委託人必須使用或管理的金鑰所需的許可。例如，使用標籤來標記專案所使用的金鑰。然後授予專案團隊許可，以僅使用具有專案標籤的金鑰。
- 要謹慎地授予主體 `payment-cryptography:TagResource` 和 `payment-cryptography:UntagResource` 許可，讓其新增、編輯和刪除別名。當您使用標籤來控制對金鑰的存取時，變更標籤可以授予主體使用他們沒有其他許可的金鑰的許可。它也可以拒絕存取其他主體執行其任務所需的金鑰。沒有變更金鑰政策或建立授予許可許可的金鑰管理員，如果擁有管理標籤的許可，則可以控制對金鑰的存取。

  盡可能使用政策條件，例如 `aws:RequestTag/`*`tag-key`*或 `aws:TagKeys`，將主體的標記許可限制為特定標籤或特定金鑰上的標籤模式。
- 檢閱您 中目前具有標記和取消標記許可 AWS 帳戶 的主體，並視需要進行調整。IAM 政策可能會允許在所有金鑰上標記和取消標記許可。例如，管理員受管政策允許主體在所有金鑰上標記、取消標記和列出標籤。
- 在設定取決於標籤的政策之前，請檢閱 中金鑰上的標籤 AWS 帳戶。請確定您的政策僅適用於您想要包含的標籤。使用CloudTrail 日誌和 CloudWatch 警示來提醒您標記可能會影響金鑰存取的變更。
- 標籤型政策條件使用模式比對；其不會繫結至標籤的特定執行個體。使用標籤型條件索引鍵的政策會影響所有符合模式的新標籤和現有標籤。如果您刪除並重新建立符合政策條件的標籤，則條件會套用至新標籤，就像舊標籤一樣。

例如，請考慮下列IAM政策。它允許委託人僅在您 帳戶中屬於美國東部（維吉尼亞北部）區域且具有`"Project"="Alpha"`標籤的金鑰上呼叫解密操作。您可以將此政策連接至 Alpha 專案範例中的角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMPolicyWithResourceTag",
```

```
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:DecryptData"
      ],
      "Resource": "arn:aws::us-east-1:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Project": "Alpha"
        }
      }
    }
  ]
}
```

下列範例IAM政策允許主體使用 帳戶中的任何金鑰進行特定密碼編譯操作。但其禁止主體在具有"Type"="Reserved"標籤或無"Type"標籤的金鑰上使用這些密碼編譯操作。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IAMAllowCryptographicOperations",
      "Effect": "Allow",
      "Action": [
        "payment-cryptography:EncryptData",
        "payment-cryptography:DecryptData",
        "payment-cryptography:ReEncrypt*"
      ],
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*"
    },
    {
      "Sid": "IAMDenyOnTag",
      "Effect": "Deny",
      "Action": [
        "payment-cryptography:EncryptData",
        "payment-cryptography:DecryptData",
        "payment-cryptography:ReEncrypt*"
      ],
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Type": "Reserved"
        }
```

```
      }
    },
    {
      "Sid": "IAMDenyNoTag",
      "Effect": "Deny",
      "Action": [
        "payment-cryptography:EncryptData",
        "payment-cryptography:DecryptData",
        "payment-cryptography:ReEncrypt*"
      ],
      "Resource": "arn:aws:kms:*:111122223333:key/*",
      "Condition": {
        "Null": {
          "aws:ResourceTag/Type": "true"
        }
      }
    }
  ]
}
```

了解 AWS Payment Cryptography 金鑰的金鑰屬性

適當的金鑰管理原則是金鑰具有適當的範圍，且只能用於允許的操作。因此，某些金鑰只能使用某些金鑰模式來建立。盡可能與 TR-31 定義的可用使用模式保持一致。

雖然 AWS 付款密碼編譯會阻止您建立無效的金鑰，但為了您的方便，此處會提供有效的組合。

對稱金鑰

- TR31_B0_BASE_DERIVATION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_C0_CARD_VERIFICATION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256

  - 允許的金鑰使用模式組合：{ Encrypt = true、Decrypt = true、 Wrap = true、Unwrap = true }、{ Encrypt = true、 Wrap = true }、{ Decrypt = true、Unwrap = true }、{ NoRestrictions = true }
- TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_E1_EMV_MKEY_CONFIDENTIALITY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_E2_EMV_MKEY_INTEGRITY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_E5_EMV_MKEY_CARD_PERSONALIZATION
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_E6_EMV_MKEY_OTHER
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ DeriveKey = true }、{ NoRestrictions = true }
- TR31_K0_KEY_ENCRYPTION_KEY
  - 允許的金鑰演算法：TDES_2KEY、TDES_3KEY、AES_128、AES_192、AES_256
  - 允許的金鑰使用模式組合：{ Encrypt = true、Decrypt = true、 Wrap = true、Unwrap = true }、{ Encrypt = true、 Wrap = true }、{ Decrypt = true、Unwrap = true }、{ NoRestrictions = true }
- TR31_K1_KEY_BLOCK_PROTECTION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Encrypt = true、Decrypt = true、 Wrap = true、Unwrap = true }、{ Encrypt = true、 Wrap = true }、{ Decrypt = true、Unwrap = true }、{ NoRestrictions = true }
- TR31_M1_ISO_9797_1_MAC_KEY
  - 允許的金鑰演算法：TDES_2KEY ，TDES_3KEY

  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_M3_ISO_9797_3_MAC_KEY
  - 允許的金鑰演算法：TDES_2KEY ，TDES_3KEY
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_M6_ISO_9797_5_CMAC_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_M7_HMAC_KEY
  - 允許的金鑰演算法：TDES_2KEY、TDES_3KEY、AES_128、AES_192、AES_256
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_P0_PIN_ENCRYPTION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Encrypt = true、Decrypt = true、 Wrap = true、Unwrap = true }、{ Encrypt = true、 Wrap = true }、{ Decrypt = true、Unwrap = true }、{ NoRestrictions = true }
- TR31_V1_IBM3624_PIN_VERIFICATION_KEY
  - 允許的金鑰演算法：TDES_2KEY 、TDES_3KEY 、AES_128 、AES_192 、AES_256
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }
- TR31_V2_VISA_PIN_VERIFICATION_KEY
  - 允許的金鑰演算法：TDES_2KEY、TDES_3KEY、AES_128、AES_192、AES_256
  - 允許的金鑰使用模式組合：{ Generate = true } ，{ Verify = true } ，{ Generate = true， Verify= true } ，{ NoRestrictions = true }

非對稱金鑰

- TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION
  - 允許的金鑰演算法：RSA_2048 、RSA_3072 、RSA_4096

  - 允許的金鑰使用模式組合：{ Encrypt = true、Decrypt = true、 Wrap = true、Unwrap = true }、{ Encrypt = true、 Wrap = true }、{ Decrypt = true、Unwrap = true }
  - NOTE：{ Encrypt = true， Wrap = true } 是匯入用於加密資料或包裝金鑰的公有金鑰時的唯一有效選項
- TR31_S0_ASYMMETRIC_KEY_FOR_DIGITAL_SIGNATURE
  - 允許的金鑰演算法：RSA_2048 、RSA_3072 、RSA_4096
  - 允許的金鑰使用模式組合：{ Sign = true } ，{ Verify = true }
  - NOTE： ：{ Verify = true } 是匯入用於簽署的金鑰時的唯一有效選項，例如根憑證、中繼憑證或 TR-34 的簽署憑證。

資料作業

建立 AWS 付款密碼編譯金鑰之後，就可以用來執行加密作業。不同的操作執行不同類型的活動，包括加密，哈希以及域特定的算法，例如 CVV2 生成。

如果沒有匹配的解密密鑰（對稱密鑰或私鑰取決於加密類型），則無法解密加密數據。如果沒有對稱密鑰或公鑰，哈希和域特定的 algorithims 也無法驗證。

如需特定作業之有效金鑰類型的資訊，請參閱密碼編譯作業的有效金鑰

> **Note**
>
> 我們建議在非生產環境中使用測試資料。在非生產環境中使用生產金鑰和資料 (PAN、BDK ID 等) 可能會影響您的合規範圍，例如 PCI DSS 和 PCI P2PE。

主題

加密，解密和重新加密數據

加密和解密方法可用於使用各種對稱和非對稱技術（包括 TDES，AES 和 RSA）來加密或解密數據。這些方法也支援使用 DUKPT 和 EM V 技術衍生的金鑰。對於希望在不暴露基礎數據的情況下保護新密鑰下的數據的用例，也可以使用該 ReEncrypt 命令。

> **Note**
>
> 當使用加密/解密函數時，所有輸入都被假定為 HexBinary-例如，1 的值將輸入為 31（十六進制），小寫 t 表示為 74（十六進制）。所有輸出也是十六進制。

有關所有可用選項的詳細信息，請參閱加密，解密和重新加密的 API 指南。

主題

- 加密資料
- 解密資料

加密資料

該 `Encrypt Data` API 用於使用對稱和非對稱數據加密密鑰以及 DUKPT 和 EM V 派生密鑰來加密數據。支援各種演算法和變化TDES，包括RSA和AES。

主要輸入是用於加密資料的加密金鑰、要加密的 HexBinary 格式的純文字資料和加密屬性，例如 TDES 區塊加密的初始化向量和模式。在的情況下，明文數據必須是 8 個字節的倍數TDES，16 個字節的倍數AES和密鑰的長度。RSA對稱鍵輸入（TDES，AES，DUKPT，EMV）應在輸入數據不符合這些要求的情況下進行填充。下表顯示每種索引鍵類型的最大純文字長度，以及您在中為 RSA 金鑰定義的`EncryptionAttributes`填補類型。

填充類型	RSA_2048	RSA_3072	RSA_4096
`OAEP SHA1`	428	684	940
`OAEP SHA256`	380	636	892
`OAEP SHA512`	252	508	764
`PKCS1`	488	744	1000
`None`	488	744	1000

主要輸出包括以 HexBinary 格式加密文字的加密資料，以及加密金鑰的總和檢查碼值。有關所有可用選項的詳細信息，請參閱 API 加密指南。

範例

- 使用 AES 對稱金鑰加密資料
- 使用 DUKPT 密鑰加密數據
- 使用 EMV 衍生的對稱金鑰加密資料

- 使用 RSA 金鑰加密資料

使用 AES 對稱金鑰加密資料

> **Note**
>
> 所有範例假設相關金鑰已存在。可以使用CreateKey作業建立金鑰，也可以使用ImportKey作業匯入金鑰。

Example

在這個例子中，我們將使用已經使用操作創建或使用CreateKey操作導入的對稱密鑰來加密明文數據。ImportKey對於此操作，密鑰必須 KeyModesOfUse 設置為`Encrypt`並將其 KeyUsage 設置為`TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY`。如需更多選項，請參閱密碼編譯作業的金鑰。

```
$ aws payment-cryptography-data encrypt-data --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi --plain-text
 31323334313233343132333431323334 --encryption-attributes 'Symmetric={Mode=CBC}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "71D7AE",
    "CipherText": "33612AB9D6929C3A828EB6030082B2BD"
}
```

使用 DUKPT 密鑰加密數據

Example

在這個例子中，我們將使用 DU KPT 密鑰加密明文數據。 AWS 支付密碼學支持`TDES`和 AES DUKPT 密鑰。對於此操作，密鑰必須 KeyModesOfUse 設置為`DeriveKey`並將其 KeyUsage 設置為`TR31_B0_BASE_DERIVATION_KEY`。如需更多選項，請參閱密碼編譯作業的金鑰。

```
$ aws payment-cryptography-data encrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
 --plain-text 31323334313233343132333431323334 --encryption-attributes
 'Dukpt={KeySerialNumber=FFFF9876543210E00001}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "71D7AE",
    "CipherText": "33612AB9D6929C3A828EB6030082B2BD"
}
```

使用 EMV 衍生的對稱金鑰加密資料

Example

在此範例中，我們將使用已建立的 EMV 衍生的對稱金鑰來加密純文字資料。您可以使用這樣的命令將資料傳送至 EMV 卡。對於此操作，密鑰必須 KeyModesOfUse 設置為`Derive`並將其 KeyUsage 設置為`TR31_E1_EMV_MKEY_CONFIDENTIALITY`或`TR31_E6_EMV_MKEY_OTHER`。如需詳細資訊，請參閱密碼編譯作業的金鑰。

```
$ aws payment-cryptography-data encrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
 --plain-text 33612AB9D6929C3A828EB6030082B2BD --encryption-attributes
 'Emv={MajorKeyDerivationMode=EMV_OPTION_A,PanSequenceNumber=27,PrimaryAccountNumber=1000000000
 InitializationVector=1500000000000999,Mode=CBC}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "71D7AE",
    "CipherText": "33612AB9D6929C3A828EB6030082B2BD"
}
```

使用 RSA 金鑰加密資料

Example

在這個例子中，我們將使用已使用操作導入的 RSA 公鑰加密明文數據。ImportKey對於此操作，密鑰必須 KeyModesOfUse 設置為`Encrypt`並將其 KeyUsage 設置為`TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION`。如需更多選項，請參閱密碼編譯作業的金鑰。

對於 PKCS #7 或其他當前不支持的填充方案，請在調用服務之前申請，並通過省略填充指示器 「不對稱 = {}」來選擇不填充

```
$ aws payment-cryptography-data encrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/thfezpmsalcfwmsg
 --plain-text 31323334313233343132333431323334 --encryption-attributes
 'Asymmetric={PaddingType=OAEP_SHA256}'
```

```
{
 "CipherText":
 "12DF6A2F64CC566D124900D68E8AFEAA794CA819876E258564D525001D00AC93047A83FB13 \
 E73F06329A100704FA484A15A49F06A7A2E55A241D276491AA91F6D2D8590C60CDE57A642BC64A897F4832A3930
 \
 0FAEC7981102CA0F7370BFBF757F271EF0BB2516007AB111060A9633D1736A9158042D30C5AE11F8C5473EC70F067
 \
 72590DEA1638E2B41FAE6FB1662258596072B13F8E2F62F5D9FAF92C12BB70F42F2ECDCF56AADF0E311D4118FE3591
 \
 FB672998CCE9D00FFFE05D2CD154E3120C5443C8CF9131C7A6A6C05F5723B8F5C07A4003A5A6173E1B425E2B5E42AD
 \
 7A2966734309387C9938B029AFB20828ACFC6D00CD1539234A4A8D9B94CDD4F23A",
 "KeyArn": "arn:aws:payment-cryptography:us-east-1:111122223333:key/5dza7xqd6soanjtb",
 "KeyCheckValue": "FF9DE9CE"
}
```

解密資料

該 `Decrypt Data` API 用於使用對稱和非對稱數據加密密鑰以及 DUKPT 和 EM V 派生密鑰來解密數據。支援各種演算法和變化TDES，包括RSA和AES。

主要輸入是用於解密數據的解密密鑰，要解密的 HexBinary 格式的密文數據和解密屬性，如初始化矢量，模式作為塊密碼等。主要輸出包括以 HexBinary 格式的純文字形式解密資料，以及解密金鑰的總和檢查碼值。有關所有可用選項的詳細信息，請參閱 API 解密指南。

範例

- 使用 AES 對稱金鑰解密資料
- 使用 DUKPT 密鑰解密數據
- 使用 EMV 衍生的對稱金鑰解密資料
- 使用 RSA 金鑰解密資料

使用 AES 對稱金鑰解密資料

Example

在這個例子中，我們將使用對稱密鑰解密密文數據。這個例子顯示了一個AES鍵TDES_2KEY，但TDES_3KEY也支持。對於此操作，密鑰必須 KeyModesOfUse 設置為Decrypt並將其 KeyUsage 設置為TR31_D0_SYMMETRIC_DATA_ENCRYPTION_KEY。如需更多選項，請參閱密碼編譯作業的金鑰。

```
$ aws payment-cryptography-data decrypt-data --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi --cipher-text
 33612AB9D6929C3A828EB6030082B2BD --decryption-attributes 'Symmetric={Mode=CBC}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "71D7AE",
    "PlainText": "31323334313233343132333431323334"
}
```

使用 DUKPT 密鑰解密數據

> **Note**
>
> 在 P2PE 交易中使用解密數據與 DUKPT 可能會將信用卡 PAN 和其他持卡人數據返回到您的應用程序，這些數據在確定其 PCI DSS 範圍時需要考慮這些數據。

Example

在這個例子中，我們將使用 DUKPT 密鑰解密密文數據，該密鑰已使用操作創建或使用CreateKey操作導入。ImportKey對於此操作，密鑰必須 KeyModesOfUse 設置為`DeriveKey`並將其 KeyUsage 設置為`TR31_B0_BASE_DERIVATION_KEY`。如需更多選項，請參閱密碼編譯作業的金鑰。當您使用時`DUKPT`，如果是`TDES`演算法，密文資料長度必須是 16 個位元組的倍數。對於`AES`演算法，密文資料長度必須是 32 個位元組的倍數。

```
$ aws payment-cryptography-data decrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
 --cipher-text 33612AB9D6929C3A828EB6030082B2BD --decryption-attributes
 'Dukpt={KeySerialNumber=FFFF9876543210E00001}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "71D7AE",
    "PlainText": "31323334313233343132333431323334"
}
```

使用 EMV 衍生的對稱金鑰解密資料

Example

在此範例中，我們將使用 EMV 衍生的對稱金鑰來解密密文資料，該金鑰是使用作業建立或使用CreateKey作業匯入的。ImportKey對於此操作，密鑰必須 KeyModesOfUse 設置為`Derive`並將其

KeyUsage 設置為TR31_E1_EMV_MKEY_CONFIDENTIALITY或TR31_E6_EMV_MKEY_OTHER。如需詳細資訊，請參閱密碼編譯作業的金鑰。

```
$ aws payment-cryptography-data decrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
 --cipher-text 33612AB9D6929C3A828EB6030082B2BD --decryption-attributes
 'Emv={MajorKeyDerivationMode=EMV_OPTION_A,PanSequenceNumber=27,PrimaryAccountNumber=1000000000
 InitializationVector=1500000000000999,Mode=CBC}'
```

```
{
"KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi",
"KeyCheckValue": "71D7AE",
"PlainText": "31323334313233343132333431323334"
}
```

使用 RSA 金鑰解密資料

Example

在這個例子中，我們將使用已使用該操作創建的 RSA 密 key pair 解密密文數據。CreateKey對於此操作，金鑰必須 KeyModesOfUse 設定為啟用Decrypt並 KeyUsage 設定為TR31_D1_ASYMMETRIC_KEY_FOR_DATA_ENCRYPTION。如需更多選項，請參閱密碼編譯作業的金鑰。

對於 PKCS #7 或其他當前不支持的填充方案，請省略填充指示器 「不對稱 = {}」 來選擇不填充，並在調用服務之後刪除填充。

```
$ aws payment-cryptography-data decrypt-data \
        --key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/5dza7xqd6soanjtb --cipher-text
 8F4C1CAFE7A5DEF9A40BEDE7F2A264635C... \
        --decryption-attributes 'Asymmetric={PaddingType=OAEP_SHA256}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-
east-1:111122223333:key/5dza7xqd6soanjtb",
    "KeyCheckValue": "FF9DE9CE",
    "PlainText": "31323334313233343132333431323334"
```

```
}
```

生成和驗證卡數據

生成和驗證卡數據包含從卡數據派生的數據，例如 CVV，CVV2，CVC 和 DCVV。

主題

- 產生卡片資料
- 驗證信用卡資料

產生卡片資料

該 `Generate Card Data` API 用於使用 CVV，CVV2 或動態 CVV2 等算法生成卡片數據。若要查看此命令可以使用哪些金鑰，請參閱密碼編譯作業的有效金鑰一節。

許多加密值，如 CVV，CVV2，ICVV，CAVV V7 使用相同的加密算法，但不同的輸入值。例如 CardVerificationValue1 具有輸入 ServiceCode，卡號和到期日期。雖然 CardVerificationValue2 只有兩個這些輸入，這是因為對於 CVV2/CVC2，則固定 ServiceCode 為 000。同樣，對於 ICVV，固定 ServiceCode 為 999。某些演算法可能會重新利用現有欄位 (例如 CAVV V8)，在這種情況下，您將需要查閱供應商手冊以取得正確的輸入值。

> **Note**
>
> 到期日必須以相同的格式 (例如 MMYY 與 YYMM) 輸入，才能產生和驗證，才能產生正確的結果。

產生 CVV2

Example

在這個例子中，我們將生成一個 CVV2 與輸入PAN和卡到期日期給定 PAN。假設您已產生卡片驗證金鑰。

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue2={CardExpiryDate=0123}
```

```
  {
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
      "KeyCheckValue": "CADDA1",
      "ValidationData": "801"
  }
```

產生 iVV

Example

在此範例中，我們將為輸入的指定 PAN 產生 ICVVPAN，服務代碼為 999 和卡片到期日。假設您已產生卡片驗證金鑰。

如需所有可用參數，請參閱 API 參考指南中的 CardVerificationValue1。

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=999}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "CADDA1",
    "ValidationData": "801"
}
```

驗證信用卡資料

`Verify Card Data`用於驗證已使用依賴加密主體的付款演算法建立的資料，例如`DISCOVER_DYNAMIC_CARD_VERIFICATION_CODE`。

輸入值通常是提供給發卡機構或支援平台合作夥伴的輸入交易的一部分。要驗證 ARQC 密碼（用於 EMV 芯片卡），請參閱驗證 ARQC。

如需詳細資訊，請參閱 API 指南VerifyCardValidationData中的。

如果該值被驗證，那麼 API 將返回 http/200。如果該值未被驗證，它將返回 http/400。

驗證

Example

在這個例子中，我們將驗證一個 CVV/CVV2 對於給定的 PAN。CVV2 通常由持卡人或用戶在交易時提供以進行驗證。為了驗證他們的輸入，下面的值將在運行時提供-密鑰用於驗證（CVK）PAN，卡到期日和 CVV2 輸入。信用卡到期日格式必須與初始值產生時使用的格式相符。

如需所有可用參數，請參閱 API 參考指南中的 CardVerificationValue2。

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue2={CardExpiryDate=0123} --validation-data 801
```

```
  {
      "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
      "KeyCheckValue": "CADDA1"
  }
```

驗證 iVV

Example

在這個例子中，我們將驗證給定 PAN 的 ICVV，輸入用於驗證的密鑰（CVK），服務代碼 999PAN，卡到期日和交易提供的 ICVV 進行驗證。

iVV 不是使用者輸入的值 (如 CVV2)，而是內嵌在 EMV 卡上。應考慮是否應該在提供時始終驗證。

如需所有可用參數，請參閱 API 參考指南中的 CardVerificationValue1。

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/tqv5yij6wtxx64pi
```

```
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=999} --validation-data 801
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
tqv5yij6wtxx64pi",
    "KeyCheckValue": "CADDA1",
    "ValidationData": "801"
}
```

生成，翻譯和驗證 PIN 數據

PIN 資料功能可讓您產生隨機 PIN 碼、PIN 碼驗證值 (PVV)，並根據 PVV 或 PIN 位移來驗證輸入的加密密碼。

引腳轉換允許您將引腳從一個工作密鑰轉換到另一個工作密鑰，而不會按照 PCI PIN 要求 1 指定的純文本顯示引腳。

> **Note**
>
> 由於 PIN 碼產生和驗證通常是發行者功能，而 PIN 碼轉譯是典型的收單機構功能，因此建議您考慮最低權限的存取權限，並針對您的系統使用案例適當地設定原則。

主題

Translate 密碼資料

轉 Translate PIN 碼資料功能可用於將加密的 PIN 資料從一組金鑰轉換為另一組金鑰，而不需要加密的資料離開 HSM。它用於 P2PE 加密，其中工作密鑰應該更改，但處理系統不需要或不允許解密數據。主要輸入是加密的數據，用於加密數據的加密密鑰，用於生成輸入值的參數。另一組輸入是請求的輸出

參數，例如用於加密輸出的密鑰以及用於創建該輸出的參數。主要輸出是新加密的資料集，以及用來產生資料集的參數。

> **Note**
>
> AES 金鑰類型僅支援 ISO 格式 4 引腳區塊。

主題

- PEK 至德國德科技試驗所的 PIN 碼
- 引腳從德科技到 AWK

PEK 至德國德科技試驗所的 PIN 碼

Example

在此示例中，我們將使用 D UKPT 算法將使用 ISO 0 密碼塊的 PEK TDES 加密的 PIN 碼轉換為 AES ISO 4 密碼塊。通常情況下，這可以反向完成，其中支付終端對 ISO 4 中的引腳進行加密，然後可以將其轉換回 TDES 以進行下游處理。

```
$ aws payment-cryptography-data translate-pin-data --encrypted-pin-block
 "AC17DC148BDA645E" --incoming-translation-
attributes=IsoFormat0='{PrimaryAccountNumber=171234567890123}' --incoming-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt  --outgoing-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/4pmyquwjs3yj4vwe --outgoing-translation-attributes
 IsoFormat4="{PrimaryAccountNumber=171234567890123}" --outgoing-dukpt-attributes
 KeySerialNumber="FFFF9876543210E00008"
```

```
    {
            "PinBlock": "1F4209C670E49F83E75CC72E81B787D9",
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
            "KeyCheckValue": "7CC9E2"
        }
```

引腳從德科技到 AWK

Example

在此範例中，我們會將 PIN 碼從 AES DUKPT 加密的 PIN 碼轉換為使用 AWK 加密的 PIN 碼。它在功能上是前一個例子的逆。

```
$ aws payment-cryptography-data translate-pin-data --encrypted-pin-
block "1F4209C670E49F83E75CC72E81B787D9" --outgoing-translation-
attributes=IsoFormat0='{PrimaryAccountNumber=171234567890123}' --outgoing-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt   --incoming-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/4pmyquwjs3yj4vwe --incoming-translation-attributes
 IsoFormat4="{PrimaryAccountNumber=171234567890123}" --incoming-dukpt-attributes
 KeySerialNumber="FFFF9876543210E00008"
```

```
        {
            "PinBlock": "AC17DC148BDA645E",
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
            "KeyCheckValue": "FE23D3"
        }
```

產生 PIN 碼資料

產生 PIN 碼資料函數用於產生 PIN 碼相關值，例如 PVV 和 PIN 區塊位移，用於在交易或授權期間驗證使用者輸入 PIN 碼。此 API 還可以使用各種算法生成新的隨機引腳。

為密碼生成簽證 PVV

Example

在此示例中，我們將生成一個新的（隨機）引腳，其中輸出將被加密`PIN block`（PinData. PinBlock) 和 a PVV (精確資料. 偏移)。關鍵輸入是`PANPin Verification Key`、、`Pin Encryption Key`和`PIN block format`.

這個命令要求密鑰的類型`TR31_V2_VISA_PIN_VERIFICATION_KEY`。

```
$ aws payment-cryptography-data generate-pin-data --generation-key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2 --encryption-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt
 --primary-account-number 171234567890123 --pin-block-format ISO_FORMAT_0 --generation-
attributes VisaPin={PinVerificationKeyIndex=1}
```

```
{
            "GenerationKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
            "GenerationKeyCheckValue": "7F2363",
            "EncryptionKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt",
            "EncryptionKeyCheckValue": "7CC9E2",
            "EncryptedPinBlock": "AC17DC148BDA645E",
            "PinData": {
                "VerificationValue": "5507"
            }
        }
```

產生針腳的 IBM3624 接腳位移

IBM 3624 PIN 碼偏移有時也稱為 IBM 方法。此方法會使用驗證資料 (通常為 PAN) 和 PIN 金鑰 (PVK) 產生自然/中間 PIN。自然引腳實際上是衍生值，並且具有確定性對於發行人來說非常有效地處理，因為不需要在持卡人級別存儲任何引腳數據。最明顯的缺點是，該方案不適用於持卡人可選擇或隨機引腳。為了允許這些類型的接腳，已在配置中加入了偏移演算法。偏移量表示使用者選取 (或隨機) 接腳與自然鍵之間的差異。偏移值由發卡機構或發卡機構儲存。在交易時， AWS 付款密碼編譯服務會在內部重新計算自然接腳，並套用偏移量以尋找接腳。然後，它將其與交易授權提供的值進行比較。

IBM3624 有幾個選項存在：

- `Ibm3624NaturalPin`將輸出自然引腳和加密的引腳塊
- `Ibm3624PinFromOffset`將生成給定偏移量的加密密腳塊
- `Ibm3624RandomPin`將生成一個隨機引腳，然後生成匹配的偏移量和加密的引腳塊。
- `Ibm3624PinOffset`在使用者選取的接腳的情況下產生接腳偏移。

內部到 AWS 付款密碼學，執行以下步驟：

- 將提供的平移填充為 16 個字符。如果提供 <16，請使用提供的填充字符在右側填充。
- 使用 PIN 產生金鑰加密驗證資料。

- 使用小數化表格將加密的資料小數化。這將十六進制數字映射到十進制數字，例如 'A' 可以映射到 9，1 可以映射到 1。
- 從輸出的十六進制表示中獲取前 4 位數字。這是天然的別針。
- 如果產生使用者選取或隨機引腳，則模數用客戶引腳減去自然引腳。結果為銷偏移。

範例

- 範例：產生接腳的 IBM3624 接腳位移

範例：產生接腳的 IBM3624 接腳位移

在此示例中，我們將生成一個新的（隨機）引腳，其中輸出將被加密PIN block（PinData. PinBlock) 和IBM3624偏移值 (精確資料. 偏移)。輸入包括PAN驗證資料 (通常是平移)、填充字元Pin Verification Key、、Pin Encryption Key和PIN block format.

此指令要求 PIN 碼產生金鑰的類型為，TR31_V1_IBM3624_PIN_VERIFICATION_KEY且加密金鑰為類型 TR31_P0_PIN_ENCRYPTION_KEY

Example

下列範例顯示產生隨機引腳，然後使用 Im3624 輸出加密的引腳區塊和 IBM3624 位移值 RandomPin

```
$ aws payment-cryptography-data generate-pin-data --generation-key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2
 --encryption-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt --primary-account-number
 171234567890123 --pin-block-format ISO_FORMAT_0 --generation-attributes
 Ibm3624RandomPin="{DecimalizationTable=9876543210654321,PinValidationDataPadCharacter=D,PinVal
```

```
{
              "GenerationKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
              "GenerationKeyCheckValue": "7F2363",
              "EncryptionKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt",
              "EncryptionKeyCheckValue": "7CC9E2",
              "EncryptedPinBlock": "AC17DC148BDA645E",
              "PinData": {
                  "PinOffset": "5507"
              }
```

```
        }
```

驗證 PIN 碼資料

驗證 PIN 數據功能用於驗證引腳是否正確。這通常涉及將先前存儲的 PIN 值與持卡人在 POI 輸入的密碼值進行比較。這些函數比較兩個值，而不會暴露任一來源的基礎值。

使用 PVV 方法驗證加密的 PIN 碼

Example

在此範例中，我們將驗證指定 PAN 的 PIN 碼。密碼通常由持卡人或用戶在交易時提供以進行驗證，並與文件中的數值進行比較（持卡人的輸入作為來自終端機或其他上游供應商的加密值提供）。為了驗證這個輸入，下面的值也將在運行時提供-用於加密輸入引腳的密鑰（這通常是一個`IWK`），以PAN及驗證的值（無論是 a PVV 或`PIN offset`）。

如果 AWS 付款密碼編譯能夠驗證引腳，則返回 http/200。如果未驗證引腳，它將返回一個 http/400。

```
$ aws payment-cryptography-data verify-pin-data --verification-key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2 --encryption-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt
 --primary-account-number 171234567890123 --pin-block-format ISO_FORMAT_0  --
verification-attributes VisaPin="{PinVerificationKeyIndex=1,VerificationValue=5507}" --
encrypted-pin-block AC17DC148BDA645E
```

```
        {
            "VerificationKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
            "VerificationKeyCheckValue": "7F2363",
            "EncryptionKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt",
            "EncryptionKeyCheckValue": "7CC9E2",

        }
```

根據先前儲存的 IBM3624 引腳位移來驗證 PIN 碼

在此範例中，我們將根據發卡機構/處理器所記錄的 PIN 碼偏移量來驗證持卡人提供的 PIN 碼。輸入類似於???支付終端（或其他上游提供商，例如卡網絡）提供的加密密碼的附加密碼。如果引腳匹

配，api 將返回 http 200。其中輸出將被加密`PIN block`（PinData. PinBlock) 和`IBM3624`偏移值 (精確資料. 偏移)。

此指令要求 PIN 碼產生金鑰的類型為，`TR31_V1_IBM3624_PIN_VERIFICATION_KEY`且加密金鑰為類型 `TR31_P0_PIN_ENCRYPTION_KEY`

Example

```
$ aws payment-cryptography-data generate-pin-data --generation-key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2
 --encryption-key-identifier arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt --primary-account-number
 171234567890123 --pin-block-format ISO_FORMAT_0 --generation-attributes
 Ibm3624RandomPin="{DecimalizationTable=9876543210654321,PinValidationDataPadCharacter=D,PinVal
```

```
{
"GenerationKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
"GenerationKeyCheckValue": "7F2363",
"EncryptionKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
"EncryptionKeyCheckValue": "7CC9E2",
"EncryptedPinBlock": "AC17DC148BDA645E",
"PinData": {
  "PinOffset": "5507"
}
}
```

驗證身份驗證請求（ARQC）密碼編譯

驗證身份驗證請求加密 API 用於驗證 AR QC。ARQC 的生成不在 AWS 支付密碼學的範圍內，通常在交易授權時間內在 EMV 芯片卡（或數字等效信息，例如移動錢包）上進行。ARQC 對每筆交易都是唯一的，旨在以密碼方式顯示卡的有效性，以及確保交易數據與當前（預期）交易完全匹配。

AWS 支付密碼學提供了多種用於驗證 ARQC 和生成可選 ARPC 值的選項，包括 EMV 4.4 第 2 冊中定義的值以及 Visa 和萬事達卡使用的其他方案。如需所有可用選項的完整清單，請參閱 API 指南中的 VerifyCardValidationData 章節。

ARQC 密碼編譯通常需要以下輸入（儘管這可能因實現而異）：

- PAN-在 PrimaryAccountNumber 欄位中指定

- PAN 序列號碼（PSN）— 在字段中指定 PanSequenceNumber
- 密鑰派生方法，例如通用會話密鑰（CSK）-在 SessionKeyDerivationAttributes
- 主密鑰派生模式（例如 EMV 選項 A）-在 MajorKeyDerivationMode
- 交易資料-在欄位中指定的各種交易、終端機和卡片資料的字串，例如「金額」和「日期 TransactionData」
- 發行者主密鑰-用於導出用於保護個別交易並在字段中指定的密碼（AC）密鑰的 KeyIdentifier 主密鑰

主題

建築物交易數據

交易資料欄位的確切內容 (和順序) 會因實作和網路配置而異，但建議的最小欄位 (和串連順序) 在 EMV 4.4 第 2 冊第 8.1.1 節- 資料選取中定義。如果前三個字段是金額（17.00），其他金額（0.00）和購買國家，這將導致交易數據開始如下：

- 000000001700-金額-12 個位置隱含兩位數十進制
- 000000000000-其他金額-12 位隱含兩位數十進制
- 0124-四位數國家代碼
- 輸出 (部分) 交易資料

交易資料填充

交易數據應在發送到服務之前填充。大多數配置使用 ISO 9797 方法 2 填充，其中一個十六進制字符串附加十六進制 80 後跟 00，直到該字段是加密塊大小的倍數; 8 個字節或 16 個字符的 TDES 和 16 個字節或 32 個字符為 AES。替代方法（方法 1）並不常見，但只使用 00 作為填充字符。

方法 1 填充

未填充字元：
0000000017000000000000000084000800080000840160517000000000093800000B03011203 (74 個字元或 37 個位元組)

已填充：
0000000017000000000000000084000800080000840160517000000000093800000B03011203 (80 個字元或 40 個位元組)

方法二填充

未填充字元：
0000000017000000000000000084000800080000840160517000000000093800000B1F220103000000 (80 個字元或 40 個位元組)

已填充：
0000000017000000000000000084000800080000840160517000000000093800000B1F220103000000 (88 個字元或 44 個位元組)

範例

美國簽證 CVN10

Example

在這個例子中，我們將驗證使用簽證 CVN10 生成的 ARQC。

如果 AWS 支付密碼學能夠驗證 ARQC，則返回 http/200。如果 ARQC 未被驗證，它將返回一個 http/400 響應。

```
$ aws payment-cryptography-data verify-auth-request-cryptogram --auth-request-
cryptogram D791093C8A921769 \
--key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk \
--major-key-derivation-mode EMV_OPTION_A \
--transaction-data
 0000000017000000000000000084000800080000840160517000000000093800000B03011203000000 \
--session-key-derivation-attributes='{"Visa":{"PanSequenceNumber":"01" \
,"PrimaryAccountNumber":"9137631040001422"}}'
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk",
  "KeyCheckValue": "08D7B4"
}
```

CVN18 簽證及簽證 CVN22

Example

在這個例子中，我們將驗證使用簽證 CVN18 或 CVN22 生成的 ARQC。CVN18 和 CVN22 之間的密碼編譯操作相同，但交易數據中包含的數據各不相同。與 CVN10 相比，即使使用相同的輸入也會產生完全不同的密碼編譯。

如果 AWS 支付密碼學能夠驗證 ARQC，則返回 http/200。如果 ARQC 未被驗證，它將返回一個 http/400。

```
$ aws payment-cryptography-data verify-auth-request-cryptogram \
--auth-request-cryptogram 61EDCC708B4C97B4
--key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk \
--major-key-derivation-mode EMV_OPTION_A
--transaction-data
 00000000170000000000000000840008000800000840160517000000000093800000B1F22010300000000000
 \
000000000000000000000000000000000000000000000800000000000000
--session-key-derivation-attributes='{"EmvCommon":
{"ApplicationTransactionCounter":"000B", \
"PanSequenceNumber":"01","PrimaryAccountNumber":"9137631040001422"}}'
```

```
{
  "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk",
  "KeyCheckValue": "08D7B4"
}
```

產生和驗證 MAC

訊息驗證碼 (MAC) 通常用於驗證訊息的完整性 (是否已修改)。加密雜湊，例如 HMAC (雜湊型訊息驗證碼)、CBC-MAC 和 CMAC (以密碼為基礎的訊息驗證碼)，另外還利用加密技術來提供 MAC 寄件者的額外保證。HMAC 是基於散列函數，而 CMAC 是基於塊密碼。

此服務的所有 MAC 演算法都結合了加密雜湊函數和共用密鑰。他們採取消息和密鑰，如密鑰中的密鑰材料，並返回一個唯一的標籤或 mac。如果消息中甚至有一個字符發生變化，或者密鑰更改，則生成的標籤是完全不同的。通過要求密鑰，加密 MAC 還提供了真實性; 如果沒有密鑰，就不可能生成相同

的 mac。加密 MAC 有時稱為對稱簽名，因為它們的工作方式類似於數字簽名，但使用單個密鑰進行簽名和驗證。

AWS支付密碼學支持幾種類型的 MAC：

演算法

由演算法 1 表KeyUsage示

ISO9797 演算法三 (零售蘋果電腦)

由演算法 3 表KeyUsage示

五号演算法

由鍵表KeyUsage示

HMAC

由香港金鑰表示，包括港澳、沙 256、香港航空公司、沙 384 KeyUsage 及

主題

- 產生 MAC
- 驗證

產生 MAC

生成 MAC API 用於通過使用已知數據值生成 MAC（消息身份驗證代碼），用於發送和接收方之間的數據驗證來驗證卡相關數據，例如從卡磁條跟踪數據。用於生成 MAC 的數據包括消息數據，秘密 MAC 加密密鑰和 MAC 算法，以生成用於傳輸的唯一 MAC 值。MAC 的接收方將使用相同的 MAC 消息數據，MAC 加密密鑰和算法來重現另一個 MAC 值進行比較和數據身份驗證。即使消息中的一個字符更改或用於驗證的 MAC 密鑰不相同，產生的 MAC 值也不同。該應用程序接口支持多孔 MAC，HMAC 和 EMV MAC 加密密鑰進行此操作。

的輸入值message-data必須是十六進位資料。

在這個例子中，我們將使用 HMAC 算法HMAC_SHA256和 HMAC 加密密鑰生成一個 HMAC（基於哈希的消息身份驗證碼）用於卡數據身份驗證。金鑰必須 KeyUsage 設定 KeyModesOfUse 為TR31_M7_HMAC_KEY和Generate。MAC 密鑰可以通過調用AWS支付密碼來創建，也可以通過調CreateKey用導入。ImportKey

Example

```
$ aws payment-cryptography-data generate-mac \
    --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6 \
    --message-data
 "3b313038383439303031303733393431353d323430383232363030303730303033f33" \
    --generation-attributes Algorithm=HMAC_SHA256
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6,
    "KeyCheckValue": "2976E7",
    "Mac": "ED87F26E961C6D0DDB78DA5038AA2BDDEA0DCE03E5B5E96BDDD494F4A7AA470C"
}
```

驗證

驗證 MAC API 是用於驗證 MAC（消息身份驗證代碼）以進行卡相關數據身份驗證。它必須使用在生成 MAC 期間使用的相同加密密鑰來重新生成 MAC 值進行身份驗證。MAC 加密密鑰可以通過調用AWS支付密碼來創建，也可CreateKey以通過調用導入。ImportKey該應用程序接口支持多孔 MAC，HMAC 和 EMV MAC 加密密鑰進行此操作。

如果該值被驗證，則響應參數`MacDataVerificationSuccessful`將返回`Http/200`，否則`Http/400`帶有消息指示`Mac verification failed`。

在此示例中，我們將使用 HMAC 算法`HMAC_SHA256`和 HMAC 加密密鑰驗證卡數據驗證來驗證 HMAC（基於哈希的消息驗證代碼）。金鑰必須 KeyUsage 設定 KeyModesOfUse 為`TR31_M7_HMAC_KEY`和`Verify`。

Example

```
$ aws payment-cryptography-data verify-mac \
     --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6 \
     --message-data
 "3b343038383439303031303733393431353d323430383232363030303730303033f33" \
     --verification-attributes='Algorithm=HMAC_SHA256' \
     --mac ED87F26E961C6D0DDB78DA5038AA2BDDEA0DCE03E5B5E96BDDD494F4A7AA470C
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
qnobl5lghrzunce6,
    "KeyCheckValue": "2976E7",
}
```

密碼編譯作業的有效金鑰

某些按鍵只能用於特定操作。此外，某些操作可能會限制按鍵使用的按鍵模式。請參閱下表以了解允許的組合。

> **Note**
>
> 某些組合雖然允許，但可能會造成無法使用的情況，例如產生 CVV 代碼，(`generate`)但後來無法驗證它們。(`verify`)

主題

GenerateCard資料

API 端點	加密操作或算法	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
GenerateCard資料	• 安全性 _ 程式碼版本 _1 • 安全性 _ 程式碼版本 _2	三十一 _ 卡驗證密鑰	• 按鍵 • 三鍵	{生成 = 真} , {生成 = 真 , 驗證 = 真}
GenerateCard資料	• 卡驗證值 _1 • 卡驗證值 _2	三十一 _ 卡驗證密鑰	• 按鍵	{生成 = 真} , {生成 = 真 , 驗證 = 真}
GenerateCard資料	• 持卡人驗證 _ 驗證 _ 值	其它電腦 _ 電腦 _ 密鑰 _	• 按鍵	{ DeriveKey = 真}
GenerateCard資料	• 動態卡驗證碼	電腦動態數字	• 按鍵	{ DeriveKey = 真}
GenerateCard資料	• 動態卡驗證值	其它電腦 _ 電腦 _ 密鑰 _	• 按鍵	{ DeriveKey = 真}

VerifyCard資料

加密操作或算法	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
• 安全性 _ 程式碼版本 _1 • 安全性 _ 程式碼版本 _2	三十一 _ 卡驗證密鑰	• 按鍵 • 三鍵	{生成 = 真} , {生成 = 真 , 驗證 = 真}
• 卡驗證值 _1 • 卡驗證值 _2	三十一 _ 卡驗證密鑰	• 按鍵	{生成 = 真} , {生成 = 真 , 驗證 = 真}

加密操作或算法	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
• 持卡人驗證 _ 驗證 _ 值	其它電腦 _ 電腦 _ 其他	• 按鍵	{ DeriveKey = 真}
• 動態卡驗證碼	電腦動態數字	• 按鍵	{ DeriveKey = 真}
• 動態卡驗證值	其它電腦 _ 電腦 _ 其他	• 按鍵	{ DeriveKey = 真}

GeneratePinData (適用於簽證/ABA 計劃)

`VISA_PIN or VISA_PIN_VERIFICATION_VALUE`

金鑰類型	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
密碼加密金鑰	密鑰加密	• 按鍵 • 三鍵	• {加密 = 真，包裝 = 真} • {加密 = 真，解密 = 真，包裝 = 真，展開 = 真} • { NoRestrictions = 真}
PIN 碼產生金鑰	驗證密鑰	• 三鍵	• {生成 = 真} • {生成 = 真，驗證 = 真}

GeneratePinData （適用於**IBM3624**）

`IBM3624_PIN_OFFSET,IBM3624_NATURAL_PIN,IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET)`

金鑰類型	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
密碼加密金鑰	密鑰加密	• 按鍵 • 三鍵	對於 IBM3624 _ 自然 _ 接腳，IBM3624 _ 隨機 _ 接腳 _ 從 _ 位移 • {加密 = 真，包裝 = 真} • {加密 = 真，解密 = 真，包裝 = 真，展開 = 真} • { NoRestrictions = 真} 對於銷 _ 位移 • {加密 = 真，解包 = 真} • {加密 = 真，解密 = 真，包裝 = 真，展開 = 真} • { NoRestrictions = 真}
PIN 碼產生金鑰	驗證密鑰	• 三鍵	• {生成 = 真} • {生成 = 真，驗證 = 真}

VerifyPinData (適用於簽證/ABA 計劃)

`VISA_PIN`

金鑰類型	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
密碼加密金鑰	密鑰加密	• 按鍵 • 三鍵	• {解密 = 真，展開 = 真} • {加密 = 真，解密 = 真，包裝 = 真，展開 = 真} • { NoRestrictions = 真}
PIN 碼產生金鑰	驗證密鑰	• 三鍵	• {驗證 = 真} • {生成 = 真，驗證 = 真}

VerifyPinData（適用於**IBM3624**）

`IBM3624_PIN_OFFSET,IBM3624_NATURAL_PIN,IBM3624_RANDOM_PIN, IBM3624_PIN_FROM_OFFSET)`

金鑰類型	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
密碼加密金鑰	密鑰加密	• 按鍵 • 三鍵	對於 IBM3624 _ 自然 _ 接腳，IBM3624 _ 隨機 _ 接腳 _ 從 _ 位移 • {解密 = 真，展開 = 真} • {加密 = 真，解密 = 真，包裝 = 真，展開 = 真} • { NoRestrictions = 真}
密碼驗證金鑰	驗證密鑰	• 三鍵	• {驗證 = 真}

金鑰類型	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
			• {生成 = 真，驗證 = 真}

解密資料

金鑰類型	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
DUKPT	基本衍生密鑰	• 按鍵 • AES_128 • AES_192 • AES_256	• { DeriveKey = 真} • { NoRestrictions = 真}
EMV	保密機密性 其它電腦 _ 電腦 _ 其他	• 按鍵	• { DeriveKey = 真}
RSA	不對稱密鑰用於數據加密	• RSA_2048 • RSA_3072 • RSA_4096	• {解密 = 真, 解包 = 真} • {加密 = 真，包裝 = 真，解密 = 真，解包 = 真}
對稱金鑰	對稱數據加密密鑰	• 按鍵 • 三鍵 • AES_128 • AES_192 • AES_256	• {解密 = 真, 解包 = 真} • {加密 = 真，包裝 = 真，解密 = 真，解包 = 真} • { NoRestrictions = 真}

加密資料

金鑰類型	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
DUKPT	基本衍生密鑰	• 按鍵 • AES_128 • AES_192 • AES_256	• { DeriveKey = 真} • { NoRestrictions = 真}
EMV	保密機密性 其它電腦 _ 電腦 _ 其他	• 按鍵	• { DeriveKey = 真}
RSA	不對稱密鑰用於數據加密	• RSA_2048 • RSA_3072 • RSA_4096	• {加密 = 真，包 = 真} • {加密 = 真，包裝 = 真，解密 = 真，解包 = 真}
對稱金鑰	對稱數據加密密鑰	• 按鍵 • 三鍵 • AES_128 • AES_192 • AES_256	• {加密 = 真，包 = 真} • {加密 = 真，包裝 = 真，解密 = 真，解包 = 真} • { NoRestrictions = 真}

Translate 接腳資料

Direction	金鑰類型	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
傳入資料來源	DUKPT	基本衍生密鑰	• 按鍵 • AES_128 • AES_192	• { DeriveKey = 真}

Direction	金鑰類型	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
			• AES_256	• { NoRestrictions = 真}
傳入資料來源	非 DukPT（PEK、AWK、IWK 等）	密鑰加密	• 按鍵 • 三鍵 • AES_128 • AES_192 • AES_256	• {解密 = 真，展開 = 真} • {加密 = 真，解密 = 真，包裝 = 真，展開 = 真} • { NoRestrictions = 真}
輸出資料目標	DUKPT	基本衍生密鑰	• 按鍵 • AES_128 • AES_192 • AES_256	• { DeriveKey = 真} • { NoRestrictions = 真}
輸出資料目標	非杜克檢測儀 (PEK、萬瓦、AWK 等)	密鑰加密	• 按鍵 • 三鍵 • AES_128 • AES_192 • AES_256	• {加密 = 真，包裝 = 真} • {加密 = 真，解密 = 真，包裝 = 真，展開 = 真} • { NoRestrictions = 真}

生成/驗證MAC

MAC 密鑰用於創建消息/數據體的加密哈希值。不建議使用有限的按鍵模式創建按鍵，因為您將無法執行匹配操作。但是，如果另一個系統打算執行另一半的作業配對，您可以只使用一個作業來匯入/匯出金鑰。

允許的金鑰使用	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
蘋果鍵	三十一 _ 馬克機碼	• 按鍵 • 三鍵	• {生成 = 真} • {生成 = 真，驗證 = 真} • {驗證 = 真} • {生成 = 真}
蘋果鑰匙 (零售)	三十一 _ 米 1 _ 金鑰	• 按鍵 • 三鍵	• {生成 = 真} • {生成 = 真，驗證 = 真} • {驗證 = 真} • {生成 = 真}
MAC 金鑰 (中華民國)	三十一 _ 米 6 _ 密鑰	• 按鍵 • 三鍵 • AES_128 • AES_192 • AES_256	• {生成 = 真} • {生成 = 真，驗證 = 真} • {驗證 = 真} • {生成 = 真}
MAC 金鑰	三十一鍵	• 按鍵 • 三鍵 • AES_128 • AES_192 • AES_256	• {生成 = 真} • {生成 = 真，驗證 = 真} • {驗證 = 真} • {生成 = 真}

VerifyAuthRequestCryptogram

允許的金鑰使用	電磁波選項	允許金鑰演算法	允許使用按鍵模式的組合
• 選項一個	應用程式密碼克	• 按鍵	• { DeriveKey = 真}

允許的金鑰使用	電磁波選項	允許金鑰演算法	允許使用按鍵模式的組合
• 選項 B			

Import/Export 鍵

操作類型	允許的金鑰使用	允許金鑰演算法	允許使用按鍵模式的組合
TR-31 包裝密鑰	金鑰區塊保護鍵 密鑰加密密鑰	• 按鍵 • 三鍵 • AES_128	• {加密 = 真，包裝 = 真}（僅導出） • {解密 = 真，展開 = 真}（僅導入） • {加密 = 真，解密 = 真，包裝 = 真，展開 = 真}
導入受信任的 CA	TR31_S0_ 非對稱 _ 密鑰 _ 形式 _ 數字簽名	• RSA_2048 • RSA_3072 • RSA_4096	• {驗證 = 真}
匯入非對稱式加密的公開金鑰憑證	不對稱密鑰用於數據加密	• RSA_2048 • RSA_3072 • RSA_4096	• {加密 = 真，包裝 = 真}

未使用的鍵類型

AWS 付款密碼學目前未使用以下金鑰類型

- 針腳產生器金鑰
- 不對稱密鑰對應關鍵字協議

常用案例

AWS 支付密碼學支持許多典型的支付加密操作。下列主題可做為如何在一般常見使用案例中使用這些作業的指南。如需所有命令的清單，請檢閱 AWS 付款密碼編譯 API。

主題

發行人和發行機構處理商

發行者使用案例通常由幾個部分組成。此區段依功能組織 (例如使用接腳)。在生產系統中，金鑰的範圍通常是指定的記憶卡匣，並且是在 bin 設定期間建立的，而不是內嵌，如下所示。

主題

一般函數

主題

生成一個隨機引腳和相關聯，PVV然後驗證該值

主題

- PIN使用PVV方法驗證加密

建立金鑰

若要產生隨機 PIN 碼，您需要兩個金鑰 PVV，一個用於產生 PIN 碼驗證金鑰 (PVK)，以PVV及用於加密 PIN 的 PIN 碼加密金鑰。引腳本身是在服務內部安全地隨機生成的，並且與任何一個密鑰加密無關。

PGK必須是以演算法本身為KEY基礎的演算法 TDES _2 的索引鍵。PVV一個PEK可以是 TDES _2KEY，TDES_3 KEY 或 AES _128。在這種情況下，由於PEK旨在在您的系統內部使用，所以 AES _128 將是一個不錯的選擇。如果 a 用PEK於與其他系統（例如卡網絡，獲取者ATMs）進行交換，或者作為遷移的一部分被移動，則出於兼容性原因，TDES_2 KEY 可能是更合適的選擇。

創建 PEK

```
$ aws payment-cryptography create-key \
               --exportable
               --key-attributes
 KeyAlgorithm=AES_128,KeyUsage=TR31_P0_PIN_ENCRYPTION_KEY,\
                   KeyClass=SYMMETRIC_KEY,\
                   KeyModesOfUse='{Encrypt=true,Decrypt=true,Wrap=true,Unwrap=true}' --
tags='[{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
               "Key": {
                   "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
                   "KeyAttributes": {
                       "KeyUsage": "TR31_P0_PIN_ENCRYPTION_KEY",
                       "KeyClass": "SYMMETRIC_KEY",
                       "KeyAlgorithm": "AES_128",
                       "KeyModesOfUse": {
                           "Encrypt": false,
                           "Decrypt": false,
                           "Wrap": false,
                           "Unwrap": false,
                           "Generate": true,
                           "Sign": false,
                           "Verify": true,
                           "DeriveKey": false,
```

```
                            "NoRestrictions": false
                        }
                    },
                    "KeyCheckValue": "7CC9E2",
                    "KeyCheckValueAlgorithm": "CMAC",
                    "Enabled": true,
                    "Exportable": true,
                    "KeyState": "CREATE_COMPLETE",
                    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                    "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
                    "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
                }
            }
```

請注意代表密鑰的，例如 ARN：AW：支付密碼：美國東-2：11112222333：密鑰/ivi5ksfsuplneuyt。`KeyArn`在下一步中，您需要這一點。

創建 PVK

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_V2_VISA_PIN_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyMo
  --tags='[{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
                  "Key": {
                      "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/ov6icy4ryas4zcza",
                      "KeyAttributes": {
                          "KeyUsage": "TR31_V2_VISA_PIN_VERIFICATION_KEY",
                          "KeyClass": "SYMMETRIC_KEY",
                          "KeyAlgorithm": "TDES_2KEY",
                          "KeyModesOfUse": {
                              "Encrypt": false,
                              "Decrypt": false,
                              "Wrap": false,
                              "Unwrap": false,
                              "Generate": true,
                              "Sign": false,
                              "Verify": true,
                              "DeriveKey": false,
                              "NoRestrictions": false
```

```
                        }
                    },
                    "KeyCheckValue": "51A200",
                    "KeyCheckValueAlgorithm": "ANSI_X9_24",
                    "Enabled": true,
                    "Exportable": true,
                    "KeyState": "CREATE_COMPLETE",
                    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                    "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
                    "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
                }
            }
```

請注意代表密鑰的，例如 ARN：AW：支付密碼：美國東部 2：11112222333：密鑰 / ov6icy4ryas4zcza。`KeyArn`在下一步中，您需要這一點。

生成一個隨機引腳，生成PVV並返回加密PIN和 PVV

Example

在此示例中，我們將生成一個新的（隨機）4 位數引腳，其中輸出將被加密`PIN block`（PinData. PinBlock) 和一個 PVV (pinData. VerificationValue). 關鍵輸入為 PAN `Pin Verification Key` (也稱為接腳產生金鑰)、`Pin Encryption Key`和PIN區塊格式。

這個命令要求密鑰的類型`TR31_V2_VISA_PIN_VERIFICATION_KEY`。

```
$ aws payment-cryptography-data generate-pin-data --generation-key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2 --encryption-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt
 --primary-account-number 171234567890123 --pin-block-format ISO_FORMAT_0 --generation-
attributes VisaPin={PinVerificationKeyIndex=1}
```

```
{
            "GenerationKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
            "GenerationKeyCheckValue": "7F2363",
            "EncryptionKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/ivi5ksfsuplneuyt",
            "EncryptionKeyCheckValue": "7CC9E2",
            "EncryptedPinBlock": "AC17DC148BDA645E",
            "PinData": {
                "VerificationValue": "5507"
            }
```

```
        }
```

PIN使用PVV方法驗證加密

Example

在這個例子中，我們將驗證一個PIN為給定的PAN。通PIN常由持卡人或用戶在交易時提供以進行驗證，並與文件中的值進行比較（持卡人的輸入作為來自終端或其他上游提供商的加密值提供）。為了驗證此輸入，在執行階段也會提供下列值-加密的 PIN 碼、用來加密輸入 PIN 碼的金鑰 (通常稱為 a IWK)，以PAN及要驗證的值 (a PVV 或PIN offset)。

如果 AWS 付款密碼編譯能夠驗證引腳，則返回 http/200。如果未驗證引腳，它將返回一個 http/400。

```
$ aws payment-cryptography-data verify-pin-data --verification-key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/37y2tsl45p5zjbh2 --encryption-
key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/ivi5ksfsuplneuyt
 --primary-account-number 171234567890123 --pin-block-format ISO_FORMAT_0  --
verification-attributes VisaPin="{PinVerificationKeyIndex=1,VerificationValue=5507}" --
encrypted-pin-block AC17DC148BDA645E
```

```
{
        "VerificationKeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/37y2tsl45p5zjbh2",
        "VerificationKeyCheckValue": "7F2363",
        "EncryptionKeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ivi5ksfsuplneuyt",
        "EncryptionKeyCheckValue": "7CC9E2",
}
```

CVV為指定卡片產生或驗證

CVV或者CVV1是傳統上嵌入在卡片磁條中的值。它與CVV2（持卡人可見並用於在線購買）不相同。

第一步是創建一個密鑰。在本自學課程中，您將建立CVK雙倍長度 3 DES (2 KEYTDES) 鍵。

> **Note**
>
> CVVCVV2，我CVV都使用類似的，如果不相同的算法，但改變輸入數據。所有使用相同的密鑰類型 TR31 _C0_ _ CARD VERIFICATION _，KEY但建議為每個目的使用單獨的密鑰。這些可以使用別名和/或標籤進行區分，如下例所示。

建立金鑰

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVV"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
            "Key": {
                "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
r52o3wbqxyf6qlqr",
                "KeyAttributes": {
                    "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY",
                    "KeyClass": "SYMMETRIC_KEY",
                    "KeyAlgorithm": "TDES_2KEY",
                    "KeyModesOfUse": {
                        "Encrypt": false,
                        "Decrypt": false,
                        "Wrap": false,
                        "Unwrap": false,
                        "Generate": true,
                        "Sign": false,
                        "Verify": true,
                        "DeriveKey": false,
                        "NoRestrictions": false
                    }
                },
                "KeyCheckValue": "DE89F9",
                "KeyCheckValueAlgorithm": "ANSI_X9_24",
                "Enabled": true,
                "Exportable": true,
                "KeyState": "CREATE_COMPLETE",
                "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
                "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
            }
        }
```

請注意代表密鑰的，例如 arn：AW：支付密碼：美國東部 2：111122333：鍵/r52o3wbxyf6qlqr。`KeyArn`在下一步中，您需要這一點。

產生一個 CVV

Example

在這個例子中，我們將生成一CVV個給定PAN的輸入PAN，121 的服務代碼（定義由ISO/IEC7813）和卡到期日。

如需所有可用參數，請參閱參API考指南中的 CardVerificationValue1。

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
r52o3wbqxyf6qlqr --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=121}'
```

```
                  {
                      "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/r52o3wbqxyf6qlqr",
                      "KeyCheckValue": "DE89F9",
                      "ValidationData": "801"
                  }
```

驗證 CVV

Example

在這個例子中，我們將驗證一個PAN與輸入給定的CVK，PAN，121 的服務代碼，卡到期日期和交易過程中CVV提供的驗證。CVV

有關所有可用參數，請參閱參API考指南中的 CardVerificationValue1。

> **Note**
>
> CVV不是用戶輸入的值（如CVV2），但通常嵌入在磁條上。應考慮是否應該在提供時始終驗證。

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/r52o3wbqxyf6qlqr
```

```
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=121} --validation-data 801
```

```
{
                     "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
r52o3wbqxyf6qlqr",
                     "KeyCheckValue": "DE89F9",
                     "ValidationData": "801"
}
```

CVV2為特定卡片產生或驗證

CVV2是傳統上在卡片背面提供的值，用於在線購物。對於虛擬卡，它可能也會顯示在應用程序或屏幕上。密碼編譯，它與CVV1不同的服務代碼值相同。

建立金鑰

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVV2"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
               "Key": {
                   "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/7f7g4spf3xcklhzu",
                   "KeyAttributes": {
                       "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY",
                       "KeyClass": "SYMMETRIC_KEY",
                       "KeyAlgorithm": "TDES_2KEY",
                       "KeyModesOfUse": {
                           "Encrypt": false,
                           "Decrypt": false,
                           "Wrap": false,
                           "Unwrap": false,
                           "Generate": true,
                           "Sign": false,
                           "Verify": true,
                           "DeriveKey": false,
```

```
                            "NoRestrictions": false
                        }
                    },
                    "KeyCheckValue": "AEA5CD",
                    "KeyCheckValueAlgorithm": "ANSI_X9_24",
                    "Enabled": true,
                    "Exportable": true,
                    "KeyState": "CREATE_COMPLETE",
                    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                    "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
                    "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
                }
            }
```

請注意代表密鑰的，例如 arn：AW：支付密碼：美國東部 2：11112222333：鍵/7f7g4spf3xcklhzu。`KeyArn`在下一步中，您需要這一點。

產生一個 CVV2

Example

在這個例子中，我們將生成一CVV2個PAN與輸入PAN和卡到期日期給定。

如需所有可用參數，請參閱參API考指南中的 CardVerificationValue2。

```
$ aws payment-cryptography-data generate-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/7f7g4spf3xcklhzu
 --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue2='{CardExpiryDate=1127}'
```

```
                        {
                            "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/7f7g4spf3xcklhzu",
                            "KeyCheckValue": "AEA5CD",
                            "ValidationData": "321"
                        }
```

驗證 CVV2

Example

在這個例子中，我們將驗證一個CVV2為一PAN個給定的輸入CVK，PAN和卡到期日期和交易過程中CVV提供的驗證。

有關所有可用參數，請參閱參API考指南中的 CardVerificationValue2。

> **Note**
>
> CVV2和其他輸入是用戶輸入的值。因此，這不一定是這個定期無法驗證的問題的標誌。

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/7f7g4spf3xcklhzu
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue2='{CardExpiryDate=1127} --validation-data 321
```

```
{
                        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/7f7g4spf3xcklhzu",
                        "KeyCheckValue": "AEA5CD",
                        "ValidationData": "801"
   }
```

CVV為特定卡片產生或驗證 i

i CVV 使用與CVV/相同的算法，CVV2但 i CVV 嵌入在芯片卡中。它的服務代碼是 999。

建立金鑰

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"ICVV"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
```

```
                "Key": {
                    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
c7dsi763r6s7lfp3",
                    "KeyAttributes": {
                        "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY",
                        "KeyClass": "SYMMETRIC_KEY",
                        "KeyAlgorithm": "TDES_2KEY",
                        "KeyModesOfUse": {
                            "Encrypt": false,
                            "Decrypt": false,
                            "Wrap": false,
                            "Unwrap": false,
                            "Generate": true,
                            "Sign": false,
                            "Verify": true,
                            "DeriveKey": false,
                            "NoRestrictions": false
                        }
                    },
                    "KeyCheckValue": "1201FB",
                    "KeyCheckValueAlgorithm": "ANSI_X9_24",
                    "Enabled": true,
                    "Exportable": true,
                    "KeyState": "CREATE_COMPLETE",
                    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                    "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
                    "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
                }
            }
```

請注意代表密鑰的，例如 ARN：AW：支付密碼：美國東部 2：11112222333：密鑰/C7dsi763r6s7lfp3。`KeyArn`在下一步中，您需要這一點。

生成一個 i CVV

Example

在這個例子中，我們將生成一個 i CVV 給定PAN的輸入PAN，一個 999 的服務代碼（定義由ISO/IEC7813）和卡到期日。

如需所有可用參數，請參閱參API考指南中的 CardVerificationValue1。

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
```

```
c7dsi763r6s7lfp3 --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=999}'
```

```
                    {
                        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/c7dsi763r6s7lfp3",
                        "KeyCheckValue": "1201FB",
                        "ValidationData": "532"
                    }
```

驗證我 CVV

Example

對於驗證，輸入為 CVK 999 的服務代碼，卡片到期日和交易期間CVV提供的 i 進行驗證。PAN

有關所有可用參數，請參閱參API考指南中的 CardVerificationValue1。

> **Note**
>
> i 不CVV是用戶輸入的值（如CVV2），但通常嵌入在EMV/芯片卡上。應考慮是否應該在提供時始終驗證。

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/c7dsi763r6s7lfp3
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue1='{CardExpiryDate=1127,ServiceCode=999} --validation-data 532
```

```
{
                      "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/c7dsi763r6s7lfp3",
                      "KeyCheckValue": "1201FB",
                      "ValidationData": "532"
   }
```

驗證EMVARQC並生成 ARPC

ARQC（授權請求加密）是由EMV（芯片）卡生成的密碼編譯，用於驗證交易詳細信息以及授權卡的使用。它結合了來自卡，終端和交易本身的數據。

在後端的驗證時，相同的輸入被提供給 AWS 付款密碼學，密碼編譯在內部重新創建，並將其與交易提供的值進行比較。從這個意義上講，它類似於MAC. EMV4.4 書 2 定義了這個函數的三個方面-密鑰派生方法（稱為通用會話密鑰-CSK）來生成一次性事務密鑰，最小有效載荷和方法用於生成響應（ARPC）。

個別信用卡方案可以指定要合併的其他交易欄位，或是這些欄位的顯示順序。其他（通常不推薦的）方案特定的派生方案也存在，並在本文檔的其他地方介紹。

若要取得更多資訊，請參閱API指南VerifyCardValidationData中的〈〉。

建立金鑰

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS,KeyClass=SYMMETRIC_KEY,KeyMod
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVN18"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
                "Key": {
                    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk",
                    "KeyAttributes": {
                        "KeyUsage": "TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS",
                        "KeyClass": "SYMMETRIC_KEY",
                        "KeyAlgorithm": "TDES_2KEY",
                        "KeyModesOfUse": {
                            "Encrypt": false,
                            "Decrypt": false,
                            "Wrap": false,
                            "Unwrap": false,
                            "Generate": false,
                            "Sign": false,
                            "Verify": false,
                            "DeriveKey": true,
                            "NoRestrictions": false
                        }
                    },
```

```
                    "KeyCheckValue": "08D7B4",
                    "KeyCheckValueAlgorithm": "ANSI_X9_24",
                    "Enabled": true,
                    "Exportable": true,
                    "KeyState": "CREATE_COMPLETE",
                    "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                    "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
                    "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
                }
            }
```

請注意代表密鑰的，例如 ARN：AW：支付密碼：美國東部 2：11112222333：密鑰/PW3s6n l62t5ushfk。`KeyArn`在下一步中，您需要這一點。

產生一個 ARQC

由EMV卡片專門產生。ARQC因此， AWS 支付密碼學沒有用於生成這樣的有效載荷的設施。出於測試目的，可以在線上使用許多庫，這些庫可以生成適當的有效負載以及各種配置通常提供的已知值。

驗證 ARQC

Example

如果 AWS 付款密碼編譯能夠驗證ARQC，則返回 http/200。ARPC(回應) 可以選擇性地提供，並在驗證之後包含在回應中。ARQC

```
$ aws payment-cryptography-data verify-auth-request-cryptogram
 --auth-request-cryptogram 61EDCC708B4C97B4 --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk
 --major-key-derivation-mode EMV_OPTION_A --transaction-data
 0000000017000000000000000084000800080000840160517000000000093800000B1F2201030000000000000000000
 --session-key-derivation-attributes='{"EmvCommon":
{"ApplicationTransactionCounter":"000B",
 "PanSequenceNumber":"01","PrimaryAccountNumber":"9137631040001422"}}' --auth-response-
attributes='{"ArpcMethod2":{"CardStatusUpdate":"12345678"}}'
```

```
{
    "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk",
    "KeyCheckValue": "08D7B4",
    "AuthResponseValue":"2263AC85"
}
```

產生並驗證 EMV MAC

EMVMAC正MAC在MAC使用EMV派生密鑰的輸入，然後對生成的數據執行 ISO9797 -3（零售）。EMVMAC通常用於將命令發送到EMV卡片，例如解除封鎖腳本。

> **Note**
>
> AWS 付款密碼編譯不驗證腳本的內容。有關要包含的特定命令的詳細信息，請參閱您的方案或卡片手冊。

若要取得更多資訊，請參閱API指南MacAlgorithmEmv中的〈〉。

主題

- 建立金鑰
- 產生一個 EMV MAC

建立金鑰

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E2_EMV_MKEY_INTEGRITY,KeyClass=SYMMETRIC_KEY,KeyModesOfUs
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVN18"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
        "Key": {
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk",
            "KeyAttributes": {
                "KeyUsage": "TR31_E2_EMV_MKEY_INTEGRITY",
                "KeyClass": "SYMMETRIC_KEY",
                "KeyAlgorithm": "TDES_2KEY",
                "KeyModesOfUse": {
                    "Encrypt": false,
                    "Decrypt": false,
                    "Wrap": false,
                    "Unwrap": false,
                    "Generate": false,
```

```
                    "Sign": false,
                    "Verify": false,
                    "DeriveKey": true,
                    "NoRestrictions": false
                }
            },
            "KeyCheckValue": "08D7B4",
            "KeyCheckValueAlgorithm": "ANSI_X9_24",
            "Enabled": true,
            "Exportable": true,
            "KeyState": "CREATE_COMPLETE",
            "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
            "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
            "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
        }
    }
```

請注意代表密鑰的，例如 ARN：AW：支付密碼：美國東部 2：11112222333：密鑰/PW3s6nl62t5ushfk。`KeyArn`在下一步中，您需要這一點。

產生一個 EMV MAC

典型的流程是後端程序會產生指令EMV碼 (例如卡解除封鎖)，使用此命令 (衍生特定於一張特定卡片的一次性金鑰) 對其進行簽署，然後傳回. MAC 然後命令 + MAC 被發送到要應用的卡。將命令發送到卡片不在 AWS 支付密碼學的範圍之外。

> **Note**
>
> 此指令適用於未傳送加密資料 (例如PIN) 時的指令。EMV加密可以與此命令結合使用，在調用此命令之前將加密的數據附加到發行者腳本

訊息資料

消息數據包括APDU標題和命令。儘管這可能因實現而異，但此示例是解除封鎖的APDU標題（84 24 00 00 08），然後按ATC（0007），然後是先前ARQC的交易（999E57FD0F47CACE）。服務不會驗證此欄位的內容。

工作階段金鑰衍生模式

此欄位定義工作階段金鑰的產生方式。EMV_ COMMON _ SESSION _ _ 通KEY常用於新的實現，而 EMV2 000 | AMEX | MASTERCARD _ SESSION _ _ KEY | 也VISA可以使用。

MajorKeyDerivationMode

EMV定義模式 A，B 或 C 模式 A 是最常見的， AWS 付款密碼學當前支持模式 A 或模式 B。

PAN

帳號，通常在晶片欄位 5A 或ISO8583欄位 2 中可用，但也可以從卡系統中取得。

PSN

卡片序號。如果未使用，請輸入 00。

SessionKeyDerivationValue

這是每個會話派生數據。它可以是 9F26 字段的最後一個ARQC（ApplicationCryptogram），也可以是 9F36 中的最後一個ATC（ ），具體取決於派生方案。

填補

填充會自動應用，並使用ISO/IEC9797-1 填充方法 2。

Example

```
$ aws payment-cryptography-data generate-mac --message-data
 84240000080007999E57FD0F47CACE --key-identifier arn:aws:payment-
cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk --message-
data 8424000008999E57FD0F47CACE0007 --generation-attributes
 EmvMac="{MajorKeyDerivationMode=EMV_OPTION_A,PanSequenceNumber='00',PrimaryAccountNumber='2235
```

```
{
"KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk",
"KeyCheckValue": "08D7B4",
"Mac":"5652EEDF83EA0D84"
}
```

網路特定功能

主題

- 簽證特定功能
- 萬事達卡特定功
- 美國運通特定功能

簽證特定功能

主題

- ARQC - CVN18/CVN22
- ARQC-CVN1 0
- CAVVV7

ARQC - CVN18/CVN22

CVN18並CVN22利用密鑰派生的CSK方法。確切的交易資料在這兩種方法之間有所不同-有關建構交易資料欄位的詳細資訊，請參閱方案文件。

ARQC-CVN1 0

CVN10 是一種較舊的 Visa 方法，用於使用每張卡密鑰推導而不是會話（每筆交EMV易）推導，並且還使用不同的有效負載的交易。有關有效載荷內容的資料，請聯絡計劃了解詳情。

建立金鑰

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS,KeyClass=SYMMETRIC_KEY,KeyMod
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVN10"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
                        "Key": {
                            "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/pw3s6nl62t5ushfk",
                            "KeyAttributes": {
                                "KeyUsage": "TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS",
                                "KeyClass": "SYMMETRIC_KEY",
                                "KeyAlgorithm": "TDES_2KEY",
                                "KeyModesOfUse": {
                                    "Encrypt": false,
                                    "Decrypt": false,
                                    "Wrap": false,
                                    "Unwrap": false,
                                    "Generate": false,
                                    "Sign": false,
                                    "Verify": false,
```

```
                                    "DeriveKey": true,
                                    "NoRestrictions": false
                                }
                            },
                            "KeyCheckValue": "08D7B4",
                            "KeyCheckValueAlgorithm": "ANSI_X9_24",
                            "Enabled": true,
                            "Exportable": true,
                            "KeyState": "CREATE_COMPLETE",
                            "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                            "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
                            "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
                        }
                    }
```

請注意代表密鑰的，例如 ARN：AW：支付密碼：美國東部 2：11112222333：密鑰/PW3s6n l62t5ushfk。`KeyArn`在下一步中，您需要這一點。

驗證 ARQC

Example

在這個例子中，我們將驗證使用 Visa CVN1 0 ARQC 生成的。

如果 AWS 付款密碼編譯能夠驗證ARQC，則返回 http/200。如果 ARQC 未被驗證，它將返回一個 http/400 響應。

```
$ aws payment-cryptography-data verify-auth-request-cryptogram --auth-request-
cryptogram D791093C8A921769 \
              --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk \
              --major-key-derivation-mode EMV_OPTION_A \
              --transaction-data
 00000000170000000000000008400080008000084016051700000000093800000B03011203000000 \
              --session-key-derivation-attributes='{"Visa":{"PanSequenceNumber":"01" \
              ,"PrimaryAccountNumber":"9137631040001422"}}'
```

```
{
                "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk",
                "KeyCheckValue": "08D7B4"
              }
```

CAVVV7

對於 Visa Secure（3DS）交易，由發行機構存取控制伺服器CAVV（ ）產生（持卡人身份驗證值 ACS）。這CAVV是發生持卡人身份驗證的證據，對於每個身份驗證交易都是唯一的，並且由收單機構在授權消息中提供。CAVVv7 將有關交易的其他數據綁定到批准中，包括商家名稱，購買金額和購買日期等要素。通過這種方式，它實際上是事務有效負載的加密哈希值。

CAVVV7 以密碼編譯方式使用該CVV算法，但輸入都已更改/重新使用。請參閱適當的第三方/簽證文件，了解如何產生輸入以產生 CAVV V7 有效載荷。

建立金鑰

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"CAVV-V7"},
{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
                    "Key": {
                        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/dnaeyrjgdjjtw6dk",
                        "KeyAttributes": {
                            "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY",
                            "KeyClass": "SYMMETRIC_KEY",
                            "KeyAlgorithm": "TDES_2KEY",
                            "KeyModesOfUse": {
                                "Encrypt": false,
                                "Decrypt": false,
                                "Wrap": false,
                                "Unwrap": false,
                                "Generate": true,
                                "Sign": false,
                                "Verify": true,
                                "DeriveKey": false,
                                "NoRestrictions": false
                            }
                        },
                        "KeyCheckValue": "F3FB13",
                        "KeyCheckValueAlgorithm": "ANSI_X9_24",
                        "Enabled": true,
```

```
                        "Exportable": true,
                        "KeyState": "CREATE_COMPLETE",
                        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                        "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
                        "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
                    }
                }
```

請注意代表密鑰的，例如 ARN：AW：支付密碼：美國東部 2：111122333：密鑰/Dnaeyrjgdjjtw6dk。`KeyArn`在下一步中，您需要這一點。

產生一個 CAVV V7

Example

在這個例子中，我們將產生一個 CAVV V7 用於在規範中指定的輸入給定的交易。請注意，對於此算法，字段可能會被重複使用/重新使用，因此不應假定字段標籤與輸入匹配。

如需所有可用參數，請參閱參API考指南中的 CardVerificationValue1。

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
dnaeyrjgdjjtw6dk --primary-account-number=171234567890123 --generation-attributes
 CardVerificationValue1='{CardExpiryDate=9431,ServiceCode=431}'
```

```
                    {
                        "KeyArn": "",
                        "KeyCheckValue": "F3FB13",
                        "ValidationData": "491"
                    }
```

驗證 7 CAVV 版

Example

對於驗證，輸入是CVK，計算的輸入值和交易過程中CAVV提供的驗證。

有關所有可用參數，請參閱參API考指南中的 CardVerificationValue1。

> **Note**
>
> CAVV不是使用者輸入的值 (如CVV2)，而是由發行者計算ACS。應考慮是否應該在提供時始終驗證。

```
$ aws payment-cryptography-data verify-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/dnaeyrjgdjjtw6dk
 --primary-account-number=171234567890123 --verification-attributes
 CardVerificationValue1='{CardExpiryDate=9431,ServiceCode=431} --validation-data 491
```

```
{
                        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/dnaeyrjgdjjtw6dk",
                        "KeyCheckValue": "F3FB13",
                        "ValidationData": "491"
    }
```

萬事達卡特定功

主題

- DCVC3
- ARQC - CVN14/CVN15
- ARQC - CVN12/CVN13

DCVC3

DCVC3早於EMVCSK和萬事達卡CVN12計劃，並代表使用動態密鑰的另一種方法。它有時也會被重新用於其他使用案例。在這種方案中，輸入是PAN，Track1/Track2 數據PSN，一個不可預知的數量和事務計數器（ ）。ATC

建立金鑰

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS,KeyClass=SYMMETRIC_KEY,KeyMod
 --tags='[{"Key":"KEY_PURPOSE","Value":"DCVC3"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
                    "Key": {
                        "KeyArn": "",
                        "KeyAttributes": {
                            "KeyUsage": "TR31_E4_EMV_MKEY_DYNAMIC_NUMBERS",
                            "KeyClass": "SYMMETRIC_KEY",
                            "KeyAlgorithm": "TDES_2KEY",
                            "KeyModesOfUse": {
                                "Encrypt": false,
                                "Decrypt": false,
                                "Wrap": false,
                                "Unwrap": false,
                                "Generate": false,
                                "Sign": false,
                                "Verify": false,
                                "DeriveKey": true,
                                "NoRestrictions": false
                            }
                        },
                        "KeyCheckValue": "",
                        "KeyCheckValueAlgorithm": "ANSI_X9_24",
                        "Enabled": true,
                        "Exportable": true,
                        "KeyState": "CREATE_COMPLETE",
                        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                        "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
                        "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
                    }
                }
```

例如，請注`KeyArn`意代表索引鍵的。在下一步中，您需要這一點。

產生一個 DCVC3

Example

雖然晶片卡可DCVC3能是由晶片卡產生，但也可以手動產生，例如此範例

```
$ aws payment-cryptography-data generate-card-validation-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk
```

```
 --primary-account-number=5413123456784808 --generation-attributes
 DynamicCardVerificationCode='{ApplicationTransactionCounter=0000,TrackData=524106000000069D13
```

```
{
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk",
            "KeyCheckValue": "08D7B4",
            "ValidationData": "865"
          }
```

驗證 DCVC3

Example

在這個例子中，我們將驗證DCVC3. 請注意，ATC應該以十六進制數字的形式提供，例如 11 的計數器應表示為 000B。該服務需要 3 位數字DCVC3，因此，如果您存儲了 4（或 5）位數值，只需截斷左邊的字符，直到有 3 位數（例如 15321 應該導致驗證數據值 321）。

如果 AWS 付款密碼編譯能夠驗證，則返回 http/200。如果該值未被驗證，它將返回一個 http/400 響應。

```
$ aws payment-cryptography-data verify-card-validation-data  --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/pw3s6nl62t5ushfk
 --primary-account-number=5413123456784808 --verification-attributes
 DynamicCardVerificationCode='{ApplicationTransactionCounter=000B,TrackData=524106000000069D13
 --validation-data 398
```

```
{
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk",
            "KeyCheckValue": "08D7B4"
          }
```

ARQC - CVN14/CVN15

CVN14並CVN15利用密鑰派生的EMVCSK方法。確切的交易資料在這兩種方法之間有所不同-有關建構交易資料欄位的詳細資訊，請參閱方案文件。

ARQC - CVN12/CVN13

CVN12並且CVN13是較舊的 Mastercard 特定於EMV交易的方法，它將不可預知的數字納入每個事務派生中並使用不同的有效負載。有關有效載荷內容的信息，請聯繫計劃。

建立金鑰

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS,KeyClass=SYMMETRIC_KEY,KeyMod
 --tags='[{"Key":"KEY_PURPOSE","Value":"CVN12"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
                    "Key": {
                        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/pw3s6nl62t5ushfk",
                        "KeyAttributes": {
                            "KeyUsage": "TR31_E0_EMV_MKEY_APP_CRYPTOGRAMS",
                            "KeyClass": "SYMMETRIC_KEY",
                            "KeyAlgorithm": "TDES_2KEY",
                            "KeyModesOfUse": {
                                "Encrypt": false,
                                "Decrypt": false,
                                "Wrap": false,
                                "Unwrap": false,
                                "Generate": false,
                                "Sign": false,
                                "Verify": false,
                                "DeriveKey": true,
                                "NoRestrictions": false
                            }
                        },
                        "KeyCheckValue": "08D7B4",
                        "KeyCheckValueAlgorithm": "ANSI_X9_24",
                        "Enabled": true,
                        "Exportable": true,
                        "KeyState": "CREATE_COMPLETE",
                        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                        "CreateTimestamp": "2024-03-07T06:41:46.648000-07:00",
                        "UsageStartTimestamp": "2024-03-07T06:41:46.626000-07:00"
                    }
                }
```

請注意代表密鑰的，例如 ARN：AW：支付密碼：美國東部 2：11112222333：密鑰/PW3s6n l62t5ushfk。`KeyArn`您需要在下一步中。

驗證 ARQC

Example

在這個例子中，我們將驗證使用萬事達卡ARQC生成的CVN12。

如果 AWS 付款密碼編譯能夠驗證ARQC，則返回 http/200。如果 ARQC 未被驗證，它將返回一個 http/400 響應。

```
$ aws payment-cryptography-data verify-auth-request-cryptogram --auth-request-
cryptogram 31BE5D49F14A5F01 \
          --key-identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk \
          --major-key-derivation-mode EMV_OPTION_A \
          --transaction-data 00000000150000000000000008400000000000008402312120197695905
 \
          --session-key-derivation-attributes='{"Mastercard":{"PanSequenceNumber":"01"
 \

 ,"PrimaryAccountNumber":"9137631040001422","ApplicationTransactionCounter":"000B","Unpredictab
```

```
{
            "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
pw3s6nl62t5ushfk",
            "KeyCheckValue": "08D7B4"
          }
```

美國運通特定功能

主題

- CSC1
- CSC2

CSC1

CSC版本 1 也被稱為經典CSC算法。該服務可以將其提供為 3,4 或 5 位數字。

如需所有可用參數，請參閱參API考指南中的 AmexCardSecurityCodeVersion1。

建立金鑰

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"CSC1"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
                    "Key": {
                        "KeyArn": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/esh6hn7pxdtttzgq",
                        "KeyAttributes": {
                            "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY",
                            "KeyClass": "SYMMETRIC_KEY",
                            "KeyAlgorithm": "TDES_2KEY",
                            "KeyModesOfUse": {
                                "Encrypt": false,
                                "Decrypt": false,
                                "Wrap": false,
                                "Unwrap": false,
                                "Generate": true,
                                "Sign": false,
                                "Verify": true,
                                "DeriveKey": false,
                                "NoRestrictions": false
                            }
                        },
                        "KeyCheckValue": "8B5077",
                        "KeyCheckValueAlgorithm": "ANSI_X9_24",
                        "Enabled": true,
                        "Exportable": true,
                        "KeyState": "CREATE_COMPLETE",
                        "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                        "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
                        "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
                    }
                }
```

注意到代表密鑰，例如 ARN：AW：支付密碼：美國東-2：11112222333：密鑰/什*KeyArn*麼 7Hn 7pxdtttzgq。您需要在下一步中。

產生一個 CSC1

Example

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
esh6hn7pxdtttzgq --primary-account-number=344131234567848 --generation-attributes
 AmexCardSecurityCodeVersion1='{CardExpiryDate=1224}' --validation-data-length 4
```

```
{
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
esh6hn7pxdtttzgq",
        "KeyCheckValue": "8B5077",
        "ValidationData": "3938"
      }
```

驗證 CSC1

Example

在這個例子中，我們將驗證一個CSC1.

如果 AWS 付款密碼編譯能夠驗證，則返回 http/200。如果該值未被驗證，它將返回一個 http/400 響應。

```
$ aws payment-cryptography-data verify-card-validation-data  --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/esh6hn7pxdtttzgq
 --primary-account-number=344131234567848 --verification-attributes
 AmexCardSecurityCodeVersion1='{CardExpiryDate=1224}''  --validation-data 3938
```

```
{
        "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
esh6hn7pxdtttzgq",
        "KeyCheckValue": "8B5077"
      }
```

CSC2

CSC版本 2 也稱為增強型CSC演算法。該服務可以將其提供為 3,4 或 5 位數字。

如需所有可用參數，請參閱參API考指南中的 AmexCardSecurityCodeVersion2。

建立金鑰

```
$ aws payment-cryptography create-key --exportable --key-attributes
 KeyAlgorithm=TDES_2KEY,KeyUsage=TR31_C0_CARD_VERIFICATION_KEY,KeyClass=SYMMETRIC_KEY,KeyModesO
 --tags='[{"Key":"KEY_PURPOSE","Value":"CSC1"},{"Key":"CARD_BIN","Value":"12345678"}]'
```

回應會回應要求參數，包括用ARN於後續呼叫以及索引鍵檢查值 (KCV)。

```
{
            "Key": {
                "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
esh6hn7pxdtttzgq",
                "KeyAttributes": {
                    "KeyUsage": "TR31_C0_CARD_VERIFICATION_KEY",
                    "KeyClass": "SYMMETRIC_KEY",
                    "KeyAlgorithm": "TDES_2KEY",
                    "KeyModesOfUse": {
                        "Encrypt": false,
                        "Decrypt": false,
                        "Wrap": false,
                        "Unwrap": false,
                        "Generate": true,
                        "Sign": false,
                        "Verify": true,
                        "DeriveKey": false,
                        "NoRestrictions": false
                    }
                },
                "KeyCheckValue": "8B5077",
                "KeyCheckValueAlgorithm": "ANSI_X9_24",
                "Enabled": true,
                "Exportable": true,
                "KeyState": "CREATE_COMPLETE",
                "KeyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                "CreateTimestamp": "2023-06-05T06:41:46.648000-07:00",
                "UsageStartTimestamp": "2023-06-05T06:41:46.626000-07:00"
            }
        }
```

注意到代表密鑰，例如 ARN：AW：支付密碼：美國東-2：11112222333：密鑰/什*KeyArn*麼 7Hn 7pxdtttzgq。您需要在下一步中。

產生一個 CSC2

在這個例子中，我們將生CSC2成一個長度為 5。CSC可以產生長度為 3,4 或 5。對於美國運通，PANs 應為 15 位數字，並以 34 或 37 開頭。

Example

```
$ aws payment-cryptography-data generate-card-validation-data --key-
identifier arn:aws:payment-cryptography:us-east-2:111122223333:key/
esh6hn7pxdtttzgq --primary-account-number=344131234567848 --generation-attributes
 AmexCardSecurityCodeVersion2='{CardExpiryDate=1224,ServiceCode=999}' --validation-
data-length 4
```

```
{
"KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/erlm445qvunmvoda",
"KeyCheckValue": "BF1077",
"ValidationData": "3982"
}
```

驗證 CSC2

Example

在這個例子中，我們將驗證一個CSC2.

如果 AWS 付款密碼編譯能夠驗證，則返回 http/200。如果該值未被驗證，它將返回一個 http/400 響應。

```
$ aws payment-cryptography-data verify-card-validation-data  --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/erlm445qvunmvoda
 --primary-account-number=344131234567848 --verification-attributes
 AmexCardSecurityCodeVersion2='{CardExpiryDate=1224,ServiceCode=999}' --validation-data
 3982
```

```
{
"KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/erlm445qvunmvoda",
"KeyCheckValue": "BF1077"
}
```

收購和支付協調員

收單銀行、PSP 和支付平台通常具有與發行機構不同的密碼編譯要求。常用案例包括：

數據解密

數據（尤其是 pan 數據）可以通過支付終端進行加密，並且需要由後端解密。解密數據和加密數據支持多種方法，包括 TDES，AES 和 DUKPT 派生技術。 AWS 付款密碼編譯服務本身也符合 PCI P2PE 標準，並註冊為 PCI P2PE 解密元件。

TranslatePin

為了維持 PCI PIN 合規性，取得系統在安全裝置上輸入持卡人 PIN 碼後，不得清楚顯示它們。因此，要將 PIN 碼從終端機傳遞到下游系統（例如支付網絡或發行商），需要使用與支付終端機所使用的密鑰不同的密鑰對其進行重新加密。Translate 引腳通過使用 Servicebbb 安全地將加密的 PIN 從一個密鑰轉換為另一個密鑰來實現這一目標。使用此命令，引腳可以在各種方案之間進行轉換，例如 TDES，AES 和 DUKPT 導出以及引腳塊格式（例如 ISO-0，ISO-3 和 ISO-4）。

VerifyMac

來自付款終端的數據可能是 MAC，以確保數據在傳輸過程中未被修改。驗證 Mac 並 GenerateMac 支持使用對稱密鑰的各種技術，包括 TDES，AES 和 DUKPT 派生技術，可與 ISO-9797-1 算法 1，ISO-9797-1 算法 3（零售 MAC）和 CMAC 技術一起使用。

其他主題

- 使用動態金鑰

使用動態金鑰

動態密鑰允許一次性或有限使用的密鑰用於加密操作，例如`EncryptData`。當關鍵材料經常旋轉（例如在每張卡片交易中），並且希望避免將關鍵材料導入服務時，可以使用此流程。短暫的金鑰可以作為軟體 POS/MP OC 或其他解決方案的一部分使用。

> **Note**
>
> 這可以用來代替使用 AWS 付款密碼編譯的典型流程，其中加密密鑰是創建或導入到服務中，並使用密鑰別名或密鑰 arn 指定密鑰。

下列操作支援動態金鑰：

- EncryptData
- DecryptData
- ReEncryptData
- TranslatePin

解密資料

下列範例顯示使用動態金鑰搭配解密鑰/解密命令。在這種情況下，金鑰識別碼是保護解密金鑰的包裝金鑰 (KEK) (在 TR-31 格式的包裝金鑰參數中提供)。包裝密鑰應 D0 的密鑰目的與解密命令以及 B 或 D 的使用模式一起使用。

Example

```
$ aws payment-cryptography-data decrypt-data --key-identifier
 arn:aws:payment-cryptography:us-east-2:111122223333:key/ov6icy4ryas4zcza
 --cipher-text 12341234123412341234123412341234123A --decryption-attributes
 'Symmetric={Mode=CBC,InitializationVector=1234123412341234}'   --wrapped-key
 WrappedKeyMaterial={"Tr31KeyBlock"="D0112D0TN00E0000B05A6E82D7FC68B95C84306634B0000DA4701BE9BC
```

```
{
   "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza",
   "KeyCheckValue": "0A3674",
   "PlainText": "2E138A746A0032023BEF5B85BA5060BA"
}
```

平移銷

下列範例顯示使用動態金鑰搭配翻譯pin 命令，將動態金鑰轉換為半靜態擷取器工作金鑰 (AWK)。在這種情況下，傳入的金鑰識別碼是保護以 TR-31 格式提供的動態 PIN 加密金鑰 (PEK) 的包裝金鑰 (KEK)。包裝密鑰應與使用 B 或 D 的模式一`P0`起的關鍵目的傳出密鑰標識符是類型的密鑰`TR31_P0_PIN_ENCRYPTION_KEY`和使用加密 = 真，WRAP = 真的模式

Example

```
$ aws payment-cryptography-data translate-pin-data --encrypted-pin-block
 "C7005A4C0FA23E02" --incoming-translation-
attributes=IsoFormat0='{PrimaryAccountNumber=171234567890123}'
```

```
 --incoming-key-identifier alias/PARTNER1_KEK  --outgoing-key-
identifier alias/ACQUIRER_AWK_PEK --outgoing-translation-attributes
 IsoFormat0="{PrimaryAccountNumber=171234567890123}"  --incoming-wrapped-key
 WrappedKeyMaterial={"Tr31KeyBlock"="D0112P0TB00S0000EB5D8E63076313162B04245C8CE351C956EA4A16CC
```

```
{
   "PinBlock": "2E66192BDA390C6F",
   "KeyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
ov6icy4ryas4zcza",
   "KeyCheckValue": "0A3674"
}
```

AWS 付款密碼編譯中的安全性

的雲端安全 AWS 是最高優先順序。身為 AWS 客戶，您受益於資料中心和網路架構，這些架構是為了滿足最安全敏感組織的需求而建置。

安全是 AWS 和 之間的共同責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 —AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。 AWS 也為您提供可安全使用的服務。第三方稽核人員會定期測試和驗證我們的安全有效性，這是AWS 合規計畫 的一部分。若要了解適用於 AWS 付款密碼編譯的合規計劃，請參閱合規計劃AWS範圍內的合規計劃。
- 雲端的安全性：您的責任取決於您使用 AWS 的服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本主題可協助您了解如何在使用 AWS Payment Cryptography 時套用共同責任模型。其中說明如何設定 AWS 付款密碼以符合您的安全和合規目標。您也會了解如何使用 AWS 其他服務來協助您監控和保護 AWS Payment Cryptography 資源。

主題

AWS Payment Cryptography 中的資料保護

AWS 共同責任模型適用於 AWS 付款密碼編譯中的資料保護。如本模型所述， AWS 負責保護執行所有 的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權。 FAQ如需歐洲資料保護的相關資訊，請參閱AWS 安全部落格 上的AWS 共同責任模型和GDPR部落格文章。

為了資料保護目的，我們建議您保護 AWS 帳戶 憑證，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management （ ） 設定個別使用者IAM。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 對每個帳戶使用多重要素驗證 （MFA）。

- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 和 建議 TLS 1.3。
- 使用 設定 API 和使用者活動記錄 AWS CloudTrail。如需使用 CloudTrail 線索擷取 AWS 活動的資訊，請參閱 AWS CloudTrail 使用者指南 中的使用 CloudTrail 線索。
- 使用 AWS 加密解決方案，以及 中的所有預設安全控制項 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在 AWS 透過命令列介面或 FIPS 存取 時需要 140-3 個經過驗證的密碼編譯模組API，請使用FIPS端點。如需可用FIPS端點的詳細資訊，請參閱聯邦資訊處理標準（FIPS）140-3。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用 AWS Payment Cryptography 或其他 AWS 服務 主控台API AWS CLI、 或 時 AWS SDKs。您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您將 URL提供給外部伺服器，強烈建議您在 中不要包含憑證資訊，URL以驗證您對該伺服器的請求。

AWS Payment Cryptography 會儲存並保護您的付款加密金鑰，使其高度可用，同時為您提供強大且靈活的存取控制。

主題

保護金鑰資料

根據預設，AWS付款密碼編譯會保護由 服務管理之付款金鑰的密碼編譯金鑰材料。此外，AWSPayment Cryptography 提供匯入在服務之外建立之金鑰材料的選項。如需付款金鑰和金鑰材料的技術詳細資訊，請參閱AWS付款密碼編譯密碼詳細資訊。

資料加密

Payment AWS Cryptography 中的資料包含 AWS Payment Cryptography 金鑰、其代表的加密金鑰材料，以及其用量屬性。金鑰材料僅以純文字存在於AWS付款密碼編譯硬體安全模組（HSMs）內，且僅在使用時存在。否則，金鑰材料和屬性會加密並存放在持久持久性持久性儲存體中。

Payment AWS Cryptography 為付款金鑰產生或載入的金鑰材料永遠不會將AWS付款密碼的界限保留HSMs在未加密的狀態。它可以由 AWS Payment Cryptography API操作進行加密匯出。

靜態加密

AWS 付款密碼編譯會在 HSM列出的 PCI PTS 中為付款金鑰產生金鑰材料HSMs。不使用時，金鑰材料會由HSM金鑰加密，並寫入持久持久的持久性儲存體。Payment Cryptography 金鑰的金鑰材料和保護金鑰材料的加密金鑰永遠不會HSMs以純文字形式保留 。

付款密碼編譯金鑰金鑰的金鑰材料加密和管理完全由 服務處理。

如需詳細資訊，請參閱 AWS Key Management Service 密碼編譯詳細資訊。

傳輸中加密

AWS Payment Cryptography 產生或載入付款金鑰的金鑰材料絕不會在AWS付款密碼編譯API操作中以純文字匯出或傳輸。AWS 付款密碼編譯使用金鑰識別符來代表API操作中的金鑰。

不過，有些 AWS Payment Cryptography API操作匯出金鑰是由先前共用或非對稱金鑰交換金鑰加密。此外，客戶可以使用 API 操作來匯入付款金鑰的加密金鑰材料。

所有AWS付款密碼編譯API呼叫都必須使用 Transport Layer Security（ ）簽署和傳輸TLS。AWS 付款密碼編譯需要 定義為PCI「強密碼編譯」的TLS版本和密碼套件。所有服務端點都支援 TLS 1.0 - 1.3 和混合後量子 TLS。

如需詳細資訊，請參閱 AWS Key Management Service 密碼編譯詳細資訊。

網際網路流量隱私權

AWS 付款密碼編譯支援AWS管理主控台和一組API操作，可讓您建立和管理付款金鑰，並在密碼編譯操作中使用它們。

AWS Pament Cryptography 支援從私有網路到 的兩個網路連線選項AWS。

- 透過網際網路的IPSecVPN連線。
- AWS Direct Connect，透過標準乙太網路光纖纜線將內部網路連結至 AWS Direct Connect 位置。

所有付款密碼編譯API呼叫都必須使用 Transport Layer Security（ ）簽署和傳輸TLS。這些呼叫還需要支援完整轉寄密碼的現代加密套件。只能透過AWS內部網路從已知的 AWS Payment Cryptography API 主機傳輸到存放付款金鑰之金鑰資料的硬體安全模組（HSMs）。

若要直接從虛擬私有雲端（VPC）連線至AWS付款密碼編譯，而不透過公有網際網路傳送流量，請使用由 支援的VPC端點AWS PrivateLink。如需詳細資訊，請參閱透過VPC端點連線至AWS付款密碼編譯。

AWS Payment Cryptography 也支援 Transport Layer Security（TLS）網路加密通訊協定的混合後總金鑰交換選項。當您連線至 AWS Payment Cryptography API端點TLS時，可以使用此選項搭配 使用。

AWS 支付密碼編譯中的復原能力

AWS 全域基礎設施是以 AWS 區域和可用區域為基礎。區域提供多個分開且隔離的實際可用區域，並以低延遲、高輸送量和高度備援網路連線相互連結。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需 AWS 區域和可用區域的詳細資訊，請參閱AWS 全域基礎設施。

區域隔離

AWS 付款密碼編譯是一項區域服務，可在多個區域使用。

區域隔離的AWS付款密碼編譯設計可確保一個AWS區域中的可用性問題不會影響任何其他區域中的AWS付款密碼編譯操作。AWS Payment Cryptography 旨在確保零計劃停機時間，並順暢且無法察覺地執行所有軟體更新和擴展操作。

AWS 付款密碼編譯服務等級協議（SLA）包含所有付款密碼編譯 99.99% 的服務承諾APIs。為了履行此承諾，AWSPayment Cryptography 可確保執行API請求所需的所有資料和授權資訊，都可用於接收請求的所有區域主機。

AWS 付款密碼編譯基礎設施在每個區域中至少在三個可用區域（AZs）中複寫。為了確保多個主機故障不會影響 AWS Payment Cryptography 效能，AWSPayment Cryptography 旨在服務來自 AZs 區域中任何 的客戶流量。

您對付款金鑰屬性或許可所做的變更會複寫至 區域中的所有主機，以確保該區域中的任何主機都可以正確處理後續請求。使用您的付款金鑰對密碼編譯操作的請求會轉送至AWS付款密碼編譯硬體安全模組（HSMs），其中任何一個都可以使用付款金鑰執行操作。

多租用戶設計

Payment AWS Cryptography 的多租戶設計使其能夠實現可用性SLA，並維持高請求率，同時保護金鑰和資料的機密性。

部署了多個完整性強制執行機制，以確保您為密碼編譯操作指定的付款金鑰始終是使用的金鑰。

付款密碼編譯金鑰的純文字金鑰材料受到廣泛保護。金鑰材料在建立HSM後立即在 中加密，且加密的金鑰材料會立即移至安全儲存。加密的金鑰會在HSM使用時間內擷取和解密。純文字金鑰只會在完成密碼編譯操作所需的時間內保留在HSM記憶體中。純文字金鑰材料永遠不會離開 HSMs；它永遠不會寫入持久性儲存體。

如需有關 AWS Payment Cryptography 用來保護金鑰之機制的詳細資訊，請參閱 AWS Payment Cryptography Cryptographic Details。

中的基礎設施安全 AWS Payment Cryptography

作為受管服務， AWS Payment Cryptography 受到 Amazon Web Services：安全程序概觀白皮書中所述 AWS 的全球網路安全程序保護。

您可以使用 AWS 已發佈的API呼叫， AWS Payment Cryptography 透過網路存取 。用戶端必須支援 Transport Layer Security （ TLS ） 1.2 或更新版本。用戶端還必須支援具有完美前向保密性 （ PFS ） 的密碼套件，例如 Ephemeral Diffie-Hellman （ DHE ） 或 Elliptic Curve Ephemeral Diffie-Hellman （ ECDHE ）。現代系統(如 Java 7 和更新版本)大多會支援這些模式。

此外，必須使用存取金鑰 ID 和與IAM委託人相關聯的秘密存取金鑰來簽署請求。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

實體主機的隔離

Payment AWS Cryptography 使用之實體基礎設施的安全受 Amazon Web Services：安全程序概觀中的實體和環境安全一節中所述的控制所約束。您可以在上一節所列的合規報告和第三方稽核問題清單中找到更多詳細資訊。

AWS 專用 commercial-off-the-shelf HSM列出的硬體安全模組 （ ） PCI PTS 支援付款密碼編譯 HSMs。Payment AWS Cryptography 金鑰的金鑰材料只會儲存在 上的揮發性記憶體中HSMs，而且只有在 Payment Cryptography 金鑰正在使用時。HSMs 位於 Amazon 資料中心內的受存取控制機架中，可強制對任何實體存取進行雙重控制。如需有關 AWS 付款密碼編譯 操作的詳細資訊HSMs，請參閱 AWS付款密碼編譯詳細資訊。

透過VPC端點連線至 AWS 付款密碼編譯

您可以透過虛擬私有雲端 （ ） 中的私有介面端點直接連線至 AWS 付款密碼編譯VPC。當您使用介面VPC端點時，您的 VPC和 AWS Payment Cryptography 之間的通訊完全在 AWS 網路內進行。

AWS 付款密碼編譯支援 支援的 Amazon Virtual Private Cloud （AmazonVPC） 端點AWS PrivateLink。每個VPC端點都由一或多個具有私有 IP 地址的彈性網路介面 （ENIs） VPC 表示。

介面VPC端點會將您的 VPC直接連線至 AWS 付款密碼編譯，而不需要網際網路閘道、NAT裝置、VPN連線或 AWS Direct Connect 連線。您 中的執行個體VPC不需要公有 IP 地址即可與 AWS Payment Cryptography 通訊。

區域

AWS 付款密碼編譯支援所有支援AWS 付款密碼編譯 AWS 區域 的VPC端點和VPC端點政策。

主題

AWS Payment Cryptography VPC端點的考量

> **Note**
>
> 雖然VPC端點可讓您在最少一個可用區域 （AZ） 中連線至 服務，但為了高可用性和備援目的，我們建議您連線至三個可用區域。

在您設定 AWS 付款密碼編譯的介面VPC端點之前，請檢閱AWS PrivateLink 指南 中的介面端點屬性和限制主題。

AWS VPC端點的付款密碼編譯支援包括以下內容。

- 您可以使用VPC端點從 呼叫所有AWS 付款密碼編譯控制平面操作和AWS 付款密碼編譯資料平面操作VPC。
- 您可以建立連線至 AWS Payment Cryptography 區域VPC端點的介面端點。
- AWS 付款密碼編譯由控制平面和資料平面組成。您可以選擇設定一個或兩個子服務，但每個子服務都會分別設定。

- 您可以使用 AWS CloudTrail 日誌，透過VPC端點稽核您對 AWS Payment Cryptography 金鑰的使用。如需詳細資訊，請參閱 記錄您的VPC端點。

建立 AWS 付款密碼編譯的VPC端點

您可以使用 Amazon VPC主控台或 Amazon VPC 建立 AWS 付款密碼編譯的VPC端點API。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的建立介面端點。

- 若要建立 AWS 付款密碼編譯的VPC端點，請使用下列服務名稱：

```
com.amazonaws.region.payment-cryptography.controlplane
```

```
com.amazonaws.region.payment-cryptography.dataplane
```

例如，在美國西部（奧勒岡）區域（`us-west-2`），服務名稱為：

```
com.amazonaws.us-west-2.payment-cryptography.controlplane
```

```
com.amazonaws.us-west-2.payment-cryptography.dataplane
```

若要更輕鬆地使用VPC端點，您可以為VPC端點啟用私有DNS名稱。如果您選取啟用DNS名稱選項，標準 AWS 付款密碼編譯DNS主機名稱會解析為您的VPC端點。例如，`https://controlplane.payment-cryptography.us-west-2.amazonaws.com`會解析為連線至服務名稱 的VPC端點`com.amazonaws.us-west-2.payment-cryptography.controlplane`。

此選項可讓您更輕鬆地使用VPC端點。和 預設 AWS SDKs AWS CLI 使用標準 AWS Payment Cryptography DNS 主機名稱，因此您不需要在應用程式和命令URL中指定VPC端點。

如需詳細資訊，請參閱《AWS PrivateLink 指南》中的透過介面端點存取服務。

連線至 AWS Payment Cryptography VPC端點

您可以使用 AWS SDK、 AWS CLI 或 ，透過VPC端點連線至 AWS 付款密碼編譯 AWS Tools for PowerShell。若要指定VPC端點，請使用其DNS名稱。

例如，此 list-keys 命令會使用 `endpoint-url` 參數來指定VPC端點。若要使用這類命令，請將範例VPC端點 ID 取代為帳戶中的範例端點 ID。

```
$ aws payment-cryptography list-keys --endpoint-url https://
vpce-1234abcdf5678c90a-09p7654s-us-east-1a.ec2.us-east-1.vpce.amazonaws.com
```

如果您在建立VPC端點時啟用私有主機名稱，則不需要在CLI命令或應用程式組態URL中指定VPC端點。標準 AWS Payment Cryptography DNS 主機名稱會解析為您的VPC端點。根據預設， AWS CLI 和 SDKs使用此主機名稱，因此您可以開始使用 VPC端點連線到 AWS Payment Cryptography 區域端點，而不需要變更指令碼和應用程式中的任何內容。

若要使用私有主機名稱，您的 `enableDnsHostnames`和 `enableDnsSupport` 屬性VPC必須設定為 `true`。若要設定這些屬性，請使用 ModifyVpcAttribute操作。如需詳細資訊，請參閱 Amazon VPC使用者指南 中的檢視和更新 的DNS屬性VPC。

控制VPC端點的存取

若要控制 AWS 對付款密碼編譯VPC端點的存取，請將VPC端點政策連接至您的VPC端點。端點政策會判斷主體是否可以使用VPC端點來呼叫具有特定 AWS Payment Cryptography 資源的 AWS Payment Cryptography 操作。

您可以在建立VPC端點時建立端點政策，而且您可以隨時變更VPC端點政策。使用 VPC 管理主控台，或 CreateVpcEndpoint或 ModifyVpcEndpoint操作。您也可以使用 AWS CloudFormation 範本 建立和變更VPC端點政策。如需使用 VPC 管理主控台的說明，請參閱 AWS PrivateLink 指南 中的建立介面端點和修改介面端點。

主題

關於VPC端點政策

對於使用VPC端點來成功的 AWS 付款密碼編譯請求，委託人需要兩個來源的許可：

- 身分型政策必須授予主體許可，以呼叫資源上的操作（AWS 付款密碼編譯金鑰或別名）。
- VPC 端點政策必須授予主體使用端點提出請求的許可。

例如，金鑰政策可能會授予主體在特定 AWS 付款密碼編譯金鑰上呼叫 Decrypt 的許可。不過，VPC端點政策可能不允許該主體使用端點`Decrypt`呼叫該 AWS 付款密碼編譯金鑰。

或者，VPC端點政策可能允許主體使用端點StopKeyUsage呼叫某些 AWS 付款密碼編譯金鑰。但是，如果委託人沒有IAM政策的這些許可，則請求會失敗。

預設VPC端點政策

每個VPC端點都有VPC端點政策，但您不需要指定政策。如果您未指定政策，則預設端點政策會允許端點上所有資源的所有委託人進行所有操作。

不過，對於 AWS 付款密碼編譯資源，委託人也必須具有從IAM政策 呼叫操作的許可。因此，實際上，預設政策指出，如果委託人具有對資源呼叫操作的許可，則其也可以使用端點來進行呼叫。

```
{
  "Statement": [
    {
      "Action": "*",
      "Effect": "Allow",
      "Principal": "*",
      "Resource": "*"
    }
  ]
}
```

若要允許主體僅將VPC端點用於其允許操作的子集，請建立或更新VPC端點政策 。

建立VPC端點政策

VPC 端點政策會判斷主體是否具有使用VPC端點對資源執行操作的許可。對於 AWS 付款密碼編譯資源，委託人還必須具有從IAM政策 執行操作的許可。

每個VPC端點政策陳述式都需要下列元素：

- 可執行動作的委託人
- 可執行的動作
- 可在其中執行動作的資源

政策陳述式不會指定VPC端點。相反地，它適用於附加政策的任何VPC端點。如需詳細資訊，請參閱 Amazon VPC使用者指南 中的使用VPC端點控制對 服務的存取。

以下是 AWS 付款密碼編譯的VPC端點政策範例。當連接至VPC端點時，此政策允許ExampleUser使用VPC端點呼叫指定 AWS 付款密碼編譯金鑰上的指定操作。使用這類政策之前，請將範例主體和金鑰識別符取代為帳戶中的有效值。

```
{
   "Statement":[
      {
         "Sid": "AllowDecryptAndView",
         "Principal": {"AWS": "arn:aws:iam::111122223333:user/ExampleUser"},
         "Effect":"Allow",
         "Action": [
             "payment-cryptography:Decrypt",
             "payment-cryptography:GetKey",
             "payment-cryptography:ListAliases",
             "payment-cryptography:ListKeys",
             "payment-cryptography:GetAlias"
          ],
         "Resource": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
      }
   ]
}
```

AWS CloudTrail 會記錄使用VPC端點的所有操作。不過，您的 CloudTrail 日誌不包括其他帳戶中主體請求的操作，或其他帳戶中 AWS 付款密碼編譯金鑰的操作。

因此，您可能想要建立VPC端點政策，以防止外部帳戶中的主體使用VPC端點來呼叫本機帳戶中任何金鑰的任何 AWS 付款密碼編譯操作。

下列範例使用 aws：PrincipalAccount全域條件金鑰拒絕存取所有 AWS 付款密碼編譯金鑰上所有操作的所有主體，除非主體位於本機帳戶中。使用這類政策之前，請將範例帳戶 ID 取代為有效值。

```
{
  "Statement": [
    {
      "Sid": "AccessForASpecificAccount",
      "Principal": {"AWS": "*"},
      "Action": "payment-cryptography:*",
      "Effect": "Deny",
      "Resource": "arn:aws:payment-cryptography:*:111122223333:key/*",
      "Condition": {
        "StringNotEquals": {
```

```
            "aws:PrincipalAccount": "111122223333"
          }
        }
      }
    ]
}
```

檢視VPC端點政策

若要檢視VPC端點的端點政策，請使用 VPC 管理主控台或 DescribeVpcEndpoints操作。

下列 AWS CLI 命令會取得具有指定端點 ID 的VPC端點政策。

使用此命令之前，請將範例端點 ID 取代為您帳戶的有效 ID。

```
$ aws ec2 describe-vpc-endpoints \
--query 'VpcEndpoints[?VpcEndpointId==`vpce-1234abcdf5678c90a`].[PolicyDocument]'
--output text
```

在政策陳述式中使用VPC端點

當請求來自VPC或使用VPC端點時，您可以控制對 AWS Payment Cryptography 資源和操作的存取。若要這麼做，請使用一個IAM政策

- 使用 `aws:sourceVpce` 條件金鑰，根據VPC端點授予或限制存取權。
- 使用 `aws:sourceVpc` 條件索引鍵，根據託管私有端點的 VPC 授予或限制存取權。

> **Note**
>
> 當請求來自 Amazon VPC端點 時，`aws:sourceIP`條件索引鍵無效。若要限制對VPC端點的請求，請使用 `aws:sourceVpce`或 `aws:sourceVpc` 條件金鑰。如需詳細資訊，請參閱AWS PrivateLink 指南 中的VPC端點和VPC端點服務的身分和存取管理。

您可以使用這些全域條件金鑰來控制對 AWS Payment Cryptography 金鑰、別名和 等操作CreateKey的存取，而這些操作不依賴於任何特定資源。

例如，以下範例金鑰政策允許使用者僅在請求使用指定的VPC端點時，使用 AWS 付款密碼編譯金鑰執行特定的密碼編譯操作，封鎖來自網際網路和連線的存取（如果設定）。當使用者向 AWS 付款密碼

編譯提出請求時，請求中的VPC端點 ID 會與政策中的`aws:sourceVpce`條件索引鍵值進行比較。如果不相符，則會拒絕請求。

若要使用像這樣的政策，請將預留位置 AWS 帳戶 ID 和VPC端點取代IDs為帳戶的有效值。

```
{
    "Id": "example-key-1",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM policies",
            "Effect": "Allow",
            "Principal": {"AWS":["111122223333"]},
            "Action": ["payment-cryptography:*"],
            "Resource": "*"
        },
        {
            "Sid": "Restrict usage to my VPC endpoint",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "payment-cryptography:Encrypt",
                "payment-cryptography:Decrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:sourceVpce": "vpce-1234abcdf5678c90a"
                }
            }
        }

    ]
}
```

您也可以使用 `aws:sourceVpc` 條件金鑰，根據VPCVPC端點所在的 限制對 AWS Payment Cryptography 金鑰的存取。

下列範例金鑰政策允許 命令，只有在金鑰來自 時才管理 AWS 付款密碼編譯金鑰`vpc-12345678`。此外，它還允許僅在密碼編譯操作使用 AWS 付款密碼編譯金鑰的命令，但僅限於來自時`vpc-2b2b2b2b`。如果應用程式正在一個 中執行VPC，但您使用第二個隔離VPC的 管理函數，則可以使用像這樣的政策。

若要使用像這樣的政策，請將預留位置 AWS 帳戶 ID 和VPC端點取代IDs為帳戶的有效值。

```
{
    "Id": "example-key-2",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow administrative actions from vpc-12345678",
            "Effect": "Allow",
            "Principal": {"AWS": "111122223333"},
            "Action": [
                "payment-cryptography:Create*","payment-
cryptography:Encrypt*","payment-cryptography:ImportKey*","payment-
cryptography:GetParametersForImport*",
                "payment-cryptography:TagResource", "payment-
cryptography:UntagResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpc": "vpc-12345678"
                }
            }
        },
        {
            "Sid": "Allow key usage from vpc-2b2b2b2b",
            "Effect": "Allow",
            "Principal": {"AWS": "111122223333"},
            "Action": [
                "payment-cryptography:Encrypt","payment-cryptography:Decrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpc": "vpc-2b2b2b2b"
                }
            }
        },
        {
            "Sid": "Allow list/read actions from everywhere",
            "Effect": "Allow",
            "Principal": {"AWS": "111122223333"},
            "Action": [
                "payment-cryptography:List*","payment-cryptography:Get*"
```

```
            ],
            "Resource": "*",
        }
    ]
}
```

記錄您的VPC端點

AWS CloudTrail 會記錄使用VPC端點的所有操作。當對 AWS Payment Cryptography 的請求使用VPC端點時，VPC端點 ID 會出現在記錄請求的AWS CloudTrail 日誌項目中。您可以使用端點 ID 稽核 AWS 付款密碼編譯VPC端點的使用。

為了保護您的 VPC，VPC端點政策 拒絕，但在其他情況下為允許請求， 不會記錄在 中AWS CloudTrail。

例如，此範例日誌項目會記錄使用VPC端點的GenerateMac請求。`vpcEndpointId` 欄位出現在日誌項目結尾。

```
{
      "eventVersion": "1.08",
      "userIdentity": {
          "principalId": "TESTXECZ5U9M4LGF2N6Y5:i-98761b8890c09a34a",
          "arn": "arn:aws:sts::111122223333:assumed-role/samplerole/
i-98761b8890c09a34a",
          "accountId": "111122223333",
          "accessKeyId": "TESTXECZ5U2ZULLHHMJG",
          "sessionContext": {
              "sessionIssuer": {
                  "type": "Role",
                  "principalId": "TESTXECZ5U9M4LGF2N6Y5",
                  "arn": "arn:aws:iam::111122223333:role/samplerole",
                  "accountId": "111122223333",
                  "userName": "samplerole"
              },
              "webIdFederationData": {},
              "attributes": {
                  "creationDate": "2024-05-27T19:34:10Z",
                  "mfaAuthenticated": "false"
              },
              "ec2RoleDelivery": "2.0"
          }
      },
      "eventTime": "2024-05-27T19:49:54Z",
```

```
      "eventSource": "payment-cryptography.amazonaws.com",
      "eventName": "CreateKey",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "172.31.85.253",
      "userAgent": "aws-cli/2.14.5 Python/3.9.16 Linux/6.1.79-99.167.amzn2023.x86_64
 source/x86_64.amzn.2023 prompt/off command/payment-cryptography.create-key",
      "requestParameters": {
          "keyAttributes": {
              "keyUsage": "TR31_M1_ISO_9797_1_MAC_KEY",
              "keyClass": "SYMMETRIC_KEY",
              "keyAlgorithm": "TDES_2KEY",
              "keyModesOfUse": {
                  "encrypt": false,
                  "decrypt": false,
                  "wrap": false,
                  "unwrap": false,
                  "generate": true,
                  "sign": false,
                  "verify": true,
                  "deriveKey": false,
                  "noRestrictions": false
              }
          },
          "exportable": true
      },
      "responseElements": {
          "key": {
              "keyArn": "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h",
              "keyAttributes": {
                  "keyUsage": "TR31_M1_ISO_9797_1_MAC_KEY",
                  "keyClass": "SYMMETRIC_KEY",
                  "keyAlgorithm": "TDES_2KEY",
                  "keyModesOfUse": {
                      "encrypt": false,
                      "decrypt": false,
                      "wrap": false,
                      "unwrap": false,
                      "generate": true,
                      "sign": false,
                      "verify": true,
                      "deriveKey": false,
                      "noRestrictions": false
                  }
```

```
                },
                "keyCheckValue": "A486ED",
                "keyCheckValueAlgorithm": "ANSI_X9_24",
                "enabled": true,
                "exportable": true,
                "keyState": "CREATE_COMPLETE",
                "keyOrigin": "AWS_PAYMENT_CRYPTOGRAPHY",
                "createTimestamp": "May 27, 2024, 7:49:54 PM",
                "usageStartTimestamp": "May 27, 2024, 7:49:54 PM"
            }
        },
        "requestID": "f3020b3c-4e86-47f5-808f-14c7a4a99161",
        "eventID": "b87c3d30-f3ab-4131-87e8-bc54cfef9d29",
        "readOnly": false,
        "eventType": "AwsApiCall",
        "managementEvent": true,
        "recipientAccountId": "111122223333",
        "vpcEndpointId": "vpce-1234abcdf5678c90a",
        "eventCategory": "Management",
        "tlsDetails": {
            "tlsVersion": "TLSv1.3",
            "cipherSuite": "TLS_AES_128_GCM_SHA256",
            "clientProvidedHostHeader": "vpce-1234abcdf5678c90a-
oo28vrvr.controlplane.payment-cryptography.us-east-1.vpce.amazonaws.com"
        }
    }
```


AWS Payment Cryptography 的安全最佳實務

AWS Payment Cryptography 支援許多內建或您可以選擇性地實作的安全功能，以增強加密金鑰的保護，並確保這些金鑰用於其預期用途，包括IAM政策、一組廣泛的政策條件金鑰，以完善您的金鑰政策和IAM政策，以及內建對金鑰區塊的PCIPIN規則強制執行。

> **Important**
>
> 提供的一般準則不代表完整的安全解決方案。由於並非所有的最佳實務都適用於所有情形，因此這些實務並非為規範性的。

- 金鑰用量和使用模式： AWS 付款密碼編譯遵循並強制執行金鑰用量和使用模式限制，如 ANSI X9 TR 31-2018 可互操作安全金鑰交換金鑰區塊規格中所述，並與PCIPIN安全要求 18-3 一

致。這限制了將單一金鑰用於多種用途的能力，並以密碼方式將金鑰中繼資料（例如允許的操作）繫結至金鑰材料本身。 AWS 付款密碼編譯會自動強制執行這些限制，例如金鑰加密金鑰（TR31_K0_KEYENCRYPTION_KEY）也無法用於資料解密。如需詳細資訊，請參閱了解 AWS Payment Cryptography 金鑰的金鑰屬性。

- 限制對稱金鑰材料的共用：只與最多一個其他實體共用對稱金鑰材料（例如 Pin Encryption Keys 或 Key Encryption Keys）。如果需要將敏感資料傳輸給更多實體或合作夥伴，請建立其他金鑰。 AWS 付款密碼編譯永遠不會清楚公開對稱金鑰材料或非對稱私有金鑰材料。
- 使用別名或標籤將金鑰與特定使用案例或合作夥伴建立關聯：別名可用來輕鬆表示與金鑰相關聯的使用案例，例如別名/BIN_12345_CVK，以表示與 BIN 12345 相關聯的卡片驗證金鑰。若要提供更多彈性，請考慮建立 bin=12345、use_case=acquiing、country=us、partner=foo 等標籤。別名和標籤也可以用於限制存取權，例如在發行和取得使用案例之間強制執行存取控制。
- 練習最低權限存取：IAM可用於限制對系統而非個人的生產存取，例如禁止個別使用者建立金鑰或執行密碼編譯操作。IAM 也可用於限制存取可能不適用於您使用案例的命令和金鑰，例如限制為收單機構產生或驗證 PIN 的能力。使用最低權限存取的另一種方法是將敏感操作（例如金鑰匯入）限制在特定服務帳戶。如需範例，請參閱 AWS 付款密碼編譯身分型政策範例。

另請參閱

- AWS Payment Cryptography 的身分和存取管理
- IAM 使用者指南中的安全最佳實務 IAM

AWS 支付密碼法規遵循驗證

協力廠商稽核人員會評估 AWS 支付密碼學的安全性與合規性，作為多項 AWS 合規計畫的一部分。這些措施包括 SOC、PCI 和其他產品。

AWS 除了 PCI DSS 之外，還針對多種 PCI 標準進行了評估，付款密碼學。其中包括 PCI 密碼安全性 (PCI PIN) 和 PCI 點對點 (P2PE) 加密。 AWS Artifact 如需可用的驗證和合規性指南，請參閱。

如需特定合規計劃範圍的 AWS 服務清單，請參閱合規計劃 AWS 服務範圍內的合規計。如需一般資訊，請參閱 AWS 合規計劃 。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

使用 AWS 付款密碼編譯時，您的合規責任取決於您資料的敏感度、貴公司的合規目標，以及適用的法律和法規。 AWS 提供下列資源以協助遵循法規：

- 安全性與合規快速入門指南 — 這些部署指南討論架構考量，並提供在上部署以安全性和法規遵循為重點的基準環境的步驟。 AWS
- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- 使用AWS Config 開發人員指南中的規則評估資源 —AWS Config；評估您的資源配置如何符合內部實踐，業界準則和法規。
- AWS Security Hub— 此 AWS 服務提供安全狀態的全面檢視，協助您檢查您 AWS 是否符合安全性產業標準和最佳做法。

AWS Payment Cryptography 的身分和存取管理

AWS Identity and Access Management （IAM） 是一種 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員會控制誰可以進行身分驗證 （登入） 和授權 （具有許可），以使用 AWS Payment Cryptography 資源。IAM 是您可以使用的 AWS 服務 ，無需額外費用。

主題

物件

使用 AWS Identity and Access Management （IAM） 的方式會有所不同，取決於您在 AWS 付款密碼編譯中執行的工作。

服務使用者 – 如果您使用 AWS Payment Cryptography 服務來執行您的工作，則您的管理員會為您提供所需的憑證和許可。當您使用更多 AWS Payment Cryptography 功能來執行工作時，您可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 AWS 付款密碼編譯中的功能，請參閱 對 AWS 付款密碼編譯身分和存取權進行疑難排解。

服務管理員 – 如果您在公司負責 AWS 付款密碼編譯資源，您可能可以完整存取 AWS 付款密碼編譯。您的任務是判斷您的服務使用者應該存取哪些 AWS 付款密碼編譯功能和資源。然後，您必須向IAM管理員提交請求，以變更服務使用者的許可。請檢閱此頁面上的資訊，以了解 的基本概念IAM。若要進一步了解貴公司如何IAM搭配 AWS Payment Cryptography 使用，請參閱 AWS 付款密碼編譯如何搭配使用 IAM。

IAM 管理員 – 如果您是IAM管理員，您可能想要了解撰寫政策以管理 AWS Payment Cryptography 存取的詳細資訊。若要檢視您可以在 中使用的以身分為基礎的 AWS Payment Cryptography 政策範例 IAM，請參閱 AWS 付款密碼編譯身分型政策範例。

使用身分驗證

驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者身分、IAM使用者身分或擔任IAM角色來驗證（登入 AWS）。

您可以使用透過身分來源提供的憑證，以聯合身分 AWS 身分登入 。 AWS IAM Identity Center（IAM Identity Center）使用者、您公司的單一登入身分驗證，以及您的 Google 或 Facebook 憑證，都是聯合身分的範例。當您以聯合身分登入時，您的管理員先前會使用 IAM角色設定身分聯合。當您 AWS 使用聯合來存取 時，您會間接擔任 角色。

您可以登入 AWS Management Console 或 AWS 存取入口網站，視您身分的使用者類型而定。如需登入 的詳細資訊 AWS，請參閱 使用者指南 中的如何登入 AWS 帳戶您的 。 AWS 登入

如果您以 AWS 程式設計方式存取 ， AWS 會提供軟體開發套件（SDK）和命令列介面（CLI），以使用您的 憑證以密碼編譯方式簽署您的請求。如果您不使用 AWS 工具，則必須自行簽署請求。如需使用建議方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南 中的 AWS 簽章版本 4 以取得API請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素身分驗證（MFA）來提高帳戶的安全性。若要進一步了解，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證，以及 IAM 使用者指南 AWS 中的多重要素驗證IAM。

AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個登入身分開始，該身分可完全存取 帳戶中的所有 AWS 服務 和資源。此身分稱為 AWS 帳戶 根使用者，透過您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需需要您以根使用者身分登入的任務完整清單，請參閱 IAM 使用者指南 中的需要根使用者憑證的任務。

IAM 使用者和群組

IAM 使用者是 中具有單一個人或應用程式特定許可 AWS 帳戶 的身分。如果可能，我們建議您依賴臨時憑證，而不是建立具有密碼和存取金鑰等長期憑證IAM的使用者。不過，如果您有特定使用案例需要IAM使用者長期憑證，建議您輪換存取金鑰。如需詳細資訊，請參閱 IAM 使用者指南 中的定期輪換需要長期憑證的使用案例存取金鑰。

IAM 群組是指定IAM使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有名為 的群組IAMAdmins，並授予該群組管理IAM資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。若要進一步了解，請參閱 IAM 使用者指南 中的IAM使用者使用案例。

IAM 角色

IAM 角色是 中具有特定許可 AWS 帳戶 的身分。它類似於IAM使用者，但與特定人員無關。若要暫時在 中擔任IAM角色 AWS Management Console，您可以從使用者切換至IAM角色（主控台）。您可以呼叫 AWS CLI 或 AWS API 操作，或使用自訂 來擔任角色URL。如需使用角色方法的詳細資訊，請參閱 IAM 使用者指南 中的擔任角色的方法。

IAM 具有臨時憑證的角色在下列情況下很有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需聯合角色的相關資訊，請參閱 IAM 使用者指南 中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，您可以設定許可集。若要控制身分在身分驗證後可以存取的內容，IAMIdentity Center 會將許可集與 中的角色相關聯IAM。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 臨時IAM使用者許可 – IAM使用者或角色可以擔任IAM角色，暫時接受特定任務的不同許可。
- 跨帳戶存取 – 您可以使用 IAM角色，允許不同帳戶中的某人（受信任的委託人）存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。不過，使用部分 AWS 服務，您可以將政策直接連接至資源（而不是使用角色作為代理）。若要了解跨帳戶存取的角色和資源型政策之間的差異，請參閱 IAM 使用者指南 中的跨帳戶資源存取IAM。
- 跨服務存取 – 有些 AWS 服務 使用其他 中的功能 AWS 服務。例如，當您在 服務中撥打電話時，該服務通常會在 Amazon 中執行應用程式EC2或在 Amazon S3 中儲存物件。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉送存取工作階段（FAS）– 當您使用IAM使用者或角色在 中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 使用呼叫 的委託人許可 AWS 服務，並結合 請求向下游服務 AWS 服務 提出請求。FAS 只有在服務收到需要與其他 AWS 服務 或 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出FAS請求的政策詳細資訊，請參閱轉送存取工作階段。
  - 服務角色 – 服務角色是服務代表您執行動作時擔任IAM的角色。IAM 管理員可以從 內部建立、修改和刪除服務角色IAM。如需詳細資訊，請參閱 使用者指南 中的建立角色以將許可委派給 AWS 服務。 IAM

- 服務連結角色 – 服務連結角色是連結至 的服務角色類型 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的 中 AWS 帳戶 ，並由 服務擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon 上執行的應用程式 EC2 – 您可以使用 IAM角色來管理在EC2執行個體上執行之應用程式的臨時憑證，以及提出 AWS CLI 或 AWS API請求。最好將存取金鑰儲存在EC2執行個體中。若要將 AWS 角色指派給EC2執行個體並將其提供給其所有應用程式，您可以建立連接至執行個體的執行個體設定檔。執行個體設定檔包含 角色，並啟用在EC2執行個體上執行的程式，以取得臨時憑證。如需詳細資訊，請參閱 IAM 使用者指南 中的使用 IAM角色將許可授予在 Amazon EC2執行個體上執行的應用程式。

使用政策管理存取權

您可以透過建立政策並將其連接至 AWS 身分或資源 AWS 來控制 中的存取。政策是 AWS 其中的物件，當與身分或資源建立關聯時， 會定義其許可。當主體 （使用者、根使用者或角色工作階段） 發出請求時， 會 AWS 評估這些政策。政策中的許可決定是否允許或拒絕請求。大多數政策都以JSON文件 AWS 形式儲存在 中。如需JSON政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南 中的JSON政策概觀。

管理員可以使用 AWS JSON政策來指定誰可以存取什麼。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對所需資源執行動作的許可，IAM管理員可以建立IAM政策。然後，管理員可以將IAM政策新增至角色，使用者可以擔任角色。

IAM 無論您用來執行操作的方法為何，政策都會定義動作的許可。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該政策的使用者可以從 AWS Management Console、 AWS CLI或 AWS 取得角色資訊API。

身分型政策

身分型政策是您可以附加到身分的JSON許可政策文件，例如IAM使用者、使用者群組或角色。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分型政策，請參閱 IAM 使用者指南 中的使用客戶受管政策定義自訂IAM許可。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管政策是獨立的政策，您可以連接到 中的多個使用者、群組和角色 AWS 帳戶。受管政策包括 AWS 受管政策和客戶受管政策。若要了解如何在受管政策或內嵌政策之間進行選擇，請參閱 IAM 使用者指南 中的在受管政策與內嵌政策之間進行選擇。

資源型政策

資源型政策是您連接至資源JSON的政策文件。資源型政策的範例包括IAM角色信任政策和 Amazon S3 儲存貯體政策 。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主體可以包括帳戶、使用者、角色、聯合使用者或 AWS 服務。

資源型政策是位於該服務中的內嵌政策。您無法在資源型政策IAM中使用來自 的 AWS 受管政策。

存取控制清單 （ACLs）

存取控制清單 （ACLs） 控制哪些主體 （帳戶成員、使用者或角色） 具有存取 資源的許可。ACLs 類似於資源型政策，雖然它們不使用JSON政策文件格式。

Amazon S3 AWS WAF和 Amazon VPC是支援 的服務範例ACLs。若要進一步了解 ACLs，請參閱 Amazon Simple Storage Service 開發人員指南 中的存取控制清單 （ACL） 概觀。

其他政策類型

AWS 支援其他較不常見的政策類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可界限是一項進階功能，您可以在其中設定身分型政策可授予IAM實體 （IAM使用者或角色） 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南 中的IAM實體許可界限。
- 服務控制政策 （SCPs） – SCPs是在 中指定組織或組織單位 （OU） 最大許可JSON的政策 AWS Organizations。 AWS Organizations 是一項用於分組和集中管理您企業擁有 AWS 帳戶 之多個的服務。如果您啟用組織中的所有功能，則可以將服務控制政策 （SCPs） 套用至任何或所有帳戶。SCP 限制成員帳戶中實體的許可，包括每個 AWS 帳戶根使用者。如需 Organizations 和 的詳細資訊SCPs，請參閱 AWS Organizations 使用者指南 中的服務控制政策。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南 中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，請參閱 IAM 使用者指南 中的政策評估邏輯。

AWS 付款密碼編譯如何搭配 使用 IAM

在您使用 IAM 管理 AWS Payment Cryptography 的存取權之前，您應該了解哪些IAM功能可與 AWS Payment Cryptography 搭配使用。若要深入了解 AWS Payment Cryptography 和其他 AWS 服務如何與 搭配使用IAM，請參閱 IAM 使用者指南 中的AWS 與 搭配使用的服務IAM。

主題

- AWS 付款密碼編譯身分型政策
- 以 AWS Payment Cryptography 標籤為基礎的授權

AWS 付款密碼編譯身分型政策

透過身分IAM型政策，您可以指定允許或拒絕的動作和資源，以及允許或拒絕動作的條件。 AWS Payment Cryptography 支援特定動作、資源和條件索引鍵。若要了解您在JSON政策中使用的所有元素，請參閱 IAM 使用者指南 中的IAMJSON政策元素參考。

動作

管理員可以使用 AWS JSON政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action`元素說明您可以用來允許或拒絕政策中存取的動作。政策動作通常具有與相關聯 AWS API操作相同的名稱。有一些例外狀況，例如沒有相符API操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

AWS Payment Cryptography 中的政策動作在動作之前使用下列字首：`payment-cryptography:`。例如，若要授予某人執行 AWS Payment Cryptography `VerifyCardData`API操作的許可，您可以在其政策中包含 `payment-cryptography:VerifyCardData`動作。政策陳述式必須包含 `Action` 或 `NotAction` 元素。 AWS Payment Cryptography 會定義自己的動作集，描述您可以使用此服務執行的任務。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示：

```
"Action": [
      "payment-cryptography:action1",
      "payment-cryptography:action2"
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定以字詞開頭的所有動作 `List`（例如 `ListKeys`和 `ListAliases`），請包含下列動作：

```
"Action": "payment-cryptography:List*"
```

若要查看 AWS 付款密碼編譯動作清單，請參閱 IAM 使用者指南 中的AWS 付款密碼編譯定義的動作。

資源

管理員可以使用 AWS JSON政策來指定誰可以存取內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Resource` JSON 政策元素會指定動作套用的物件。陳述式必須包含 `Resource` 或 `NotResource` 元素。最佳實務是使用其 Amazon Resource Name（ARN）指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

付款加密金鑰資源具有下列 ARN：

```
arn:${Partition}:payment-cryptography:${Region}:${Account}:key/${keyARN}
```

如需有關 格式的詳細資訊ARNs，請參閱 Amazon Resource Names（ARNs）AWS 和服務命名空間。

例如，若要在陳述式中指定`arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h`執行個體，請使用下列 ARN：

```
"Resource": "arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h"
```

若要指定屬於特定帳戶的所有金鑰，請使用萬用字元（*）：

```
"Resource": "arn:aws:payment-cryptography:us-east-2:111122223333:key/*"
```

某些 AWS 付款密碼編譯動作，例如用於建立金鑰的動作，無法在特定資源上執行。在這些情況下，您必須使用萬用字元 (*)。

```
"Resource": "*"
```

若要在單一陳述式中指定多個資源，請使用逗號，如下所示：

```
"Resource": [
      "resource1",
      "resource2"
```

範例

若要檢視 AWS 以 Payment Cryptography 身分為基礎的政策範例，請參閱 AWS 付款密碼編譯身分型政策範例。

以 AWS Payment Cryptography 標籤為基礎的授權

AWS 付款密碼編譯身分型政策範例

根據預設，IAM使用者和角色沒有建立或修改 AWS 付款密碼編譯資源的許可。他們也無法使用 AWS Management Console AWS CLI、 或 執行任務 AWS API。IAM 管理員必須建立IAM政策，授予使用者和角色許可，以對所需的指定資源執行特定API操作。然後，管理員必須將這些政策連接到需要這些許可IAM的使用者或群組。

若要了解如何使用這些範例政策文件建立IAM身分型JSON政策，請參閱 IAM 使用者指南 中的在 JSON 索引標籤上建立政策。

主題

- 政策最佳實務
- 使用 AWS 付款密碼編譯主控台
- 允許使用者檢視他們自己的許可

- 能夠存取 AWS 付款密碼學的所有層面
- 能夠APIs使用指定的金鑰呼叫
- 明確拒絕資源的能力

政策最佳實務

身分型政策會決定某人是否可以在帳戶中建立、存取或刪除 AWS 付款密碼編譯資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用針對許多常見使用案例授予許可的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。如需詳細資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或 AWS 受管政策。
- 套用最低權限許可 – 當您使用IAM政策設定許可時， 只會授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的詳細資訊，請參閱 IAM 使用者指南 中的政策和許可IAM。
- 使用IAM政策中的條件來進一步限制存取：您可以將條件新增至政策，以限制對動作和資源的存取。例如，您可以撰寫政策條件來指定所有請求都必須使用 傳送SSL。如果透過特定 使用服務動作，例如 AWS 服務，您也可以使用 條件來授予其存取權 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南 中的IAMJSON政策元素：條件。
- 使用 IAM Access Analyzer 驗證您的IAM政策以確保安全且功能許可 – IAM Access Analyzer 會驗證新的和現有的政策，讓政策遵循IAM政策語言 （JSON） 和IAM最佳實務。IAM Access Analyzer 提供超過 100 個政策檢查和可操作的建議，協助您撰寫安全且實用的政策。如需詳細資訊，請參閱 IAM 使用者指南 中的使用 IAM Access Analyzer 驗證政策。
- 需要多重要素身分驗證 （MFA） – 如果您有需要 IAM使用者或 根使用者的案例 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。若要在呼叫API操作MFA時要求 ，請將MFA條件新增至您的政策。如需詳細資訊，請參閱 IAM 使用者指南 中的使用 安全API存取MFA。

如需 中最佳實務的詳細資訊IAM，請參閱 IAM 使用者指南 中的安全最佳實務IAM。

使用 AWS 付款密碼編譯主控台

若要存取 AWS Payment Cryptography 主控台，您必須具有一組最低許可。這些許可必須允許您列出和檢視 AWS 帳戶中 AWS 付款密碼編譯資源的詳細資訊。如果您建立的身分型政策比最低必要許可更嚴格，則主控台將無法與該政策針對實體 （IAM使用者或角色） 正常運作。

為了確保這些實體仍然可以使用 AWS Payment Cryptography 主控台，請將下列 AWS 受管政策連接至實體。如需詳細資訊，請參閱 使用者指南 中的將許可新增至使用者。 IAM

對於僅對 AWS CLI 或 進行呼叫的使用者，您不需要允許最低主控台許可 AWS API。相反地，僅允許存取與您嘗試執行API的操作相符的動作。

允許使用者檢視他們自己的許可

此範例示範如何建立政策，允許使用者檢視連接至其IAM使用者身分的內嵌和受管政策。此政策包含在主控台上完成此動作或使用 AWS CLI 或 以程式設計方式完成此動作的許可 AWS API。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

能夠存取 AWS 付款密碼學的所有層面

> **Warning**
>
> 此範例提供廣泛的許可，不建議使用 。改為考慮最低權限的存取模型。

在此範例中，您想要授予 AWS 帳戶中IAM的使用者存取所有 AWS Payment Cryptography 金鑰，以及呼叫所有 AWS Payment Cryptography api 的功能，包括 ControlPlane 和 DataPlane 操作。

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
            "Effect": "Allow",
            "Action": [
               "payment-cryptography:*"
            ],
            "Resource": [
               "*"
            ]
         }
   ]
}
```

能夠APIs使用指定的金鑰呼叫

在此範例中，您想要授予 AWS 帳戶中IAM的使用者存取其中一個 AWS Payment Cryptography 金鑰，`arn:aws:payment-cryptography:us-east-2:111122223333:key/kwapwa6qaifllw2h`然後在兩個 APIs、 `GenerateCardData`和 中使用此資源`VerifyCardData`。相反地，IAM使用者將無法存取在其他操作上使用此金鑰，例如 `DeleteKey`或 `ExportKey`

資源可以是索引鍵、字首為 `key`或別名、字首為 `alias`。

```
{
   "Version": "2012-10-17",
   "Statement": [
         {
            "Effect": "Allow",
```

```
                "Action": [
                   "payment-cryptography:VerifyCardData",
                   "payment-cryptography:GenerateCardData"
                ],
                "Resource": [
                   "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
                ]
            }
      ]
   }
```

明確拒絕資源的能力

> **Warning**
>
> 仔細考慮授予萬用字元存取的影響。請改為考慮最低權限模型。

在此範例中，您想要允許 AWS 帳戶中IAM的使用者存取任何 AWS Payment Cryptography 金鑰，但想要拒絕某個特定金鑰的許可。除了拒絕陳述式中指定的金鑰之外，使用者將可存取 `VerifyCardData`和 `GenerateCardData`的所有金鑰。

```
   {
      "Version": "2012-10-17",
      "Statement": [
          {
              "Effect": "Allow",
              "Action": [
                  "payment-cryptography:VerifyCardData",
                  "payment-cryptography:GenerateCardData"
              ],
              "Resource": [
                  "arn:aws:payment-cryptography:us-east-2:111122223333:key/*"
              ]
          },
          {
              "Effect": "Deny",
              "Action": [
                  "payment-cryptography:GenerateCardData"
              ],
```

```
                "Resource": [
                    "arn:aws:payment-cryptography:us-east-2:111122223333:key/
kwapwa6qaifllw2h"
                ]
            }
        ]
    }
```

對 AWS 付款密碼編譯身分和存取權進行疑難排解

主題將新增至本節，因為已識別 AWS 付款密碼編譯特有的 IAM相關問題。如需IAM主題的一般疑難排解內容，請參閱 IAM 使用者指南 的疑難排解一節。

監控 AWS 付款密碼

監控是維護 AWS 付款加密和其他 AWS 解決方案的可靠性、可用性和效能的重要組成部分。AWS 提供下列監控工具來觀看 AWS 付款密碼編譯、在發生錯誤時回報，並在適當時採取自動動作：

- Amazon 會即時 CloudWatch監控您的 AWS 資源和執行 AWS 的應用程式。您可以收集和追蹤指標、建立自訂儀板表，以及設定警示，在特定指標達到您指定的閾值時通知您或採取動作。例如，您可以 CloudWatch 追蹤特定 API 的使用情況，或是在接近 AWS 付款密碼編譯配額時通知您。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。
- Amazon CloudWatch 日誌可讓您從 Amazon EC2 執行個體和其他來源監控 CloudTrail、存放和存取日誌檔。 CloudWatch 記錄檔可以監控記錄檔中的資訊，並在符合特定臨界值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南。
- AWS CloudTrail擷取您帳戶或代表您 AWS 帳戶發出的 API 呼叫和相關事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。您可以識別呼叫的使用者和帳戶 AWS、呼叫的端點、使用的資源 (金鑰)、進行呼叫的來源 IP 位址，以及呼叫發生的時間。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》。

主題

- 使用記錄 AWS 付款密碼編譯API呼叫 AWS CloudTrail

使用記錄 AWS 付款密碼編譯API呼叫 AWS CloudTrail

AWS 支付密碼學與 AWS CloudTrail，提供由用戶，角色或支 AWS 付密碼學服務採取的操作的記錄 AWS 服務集成的服務。 CloudTrail 捕獲所有 AWS 支付密碼編譯的API呼叫作為事件。擷取的呼叫包括來自主控台的呼叫和對API作業的程式碼呼叫。如果您建立追蹤，您可以啟用持續交付 CloudTrail 事件到 Amazon S3 儲存貯體，包括 AWS 付款密碼編譯的事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 的主控台中檢視最新的管理 (CloudTrail Control Plane) 事件。使用收集的資訊 CloudTrail，您可以判斷向 AWS 付款密碼學提出的要求、提出要求的 IP 位址、提出要求的人員、提出要求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail 用者指南。

主題

- AWS 付款密碼學資訊 CloudTrail

- 控制平面事件 CloudTrail
- 資料事件 CloudTrail
- 瞭解 AWS 付款密碼編譯控制平面記錄檔項目
- 瞭解 AWS 付款密碼編譯資料平面記錄檔項目

AWS 付款密碼學資訊 CloudTrail

CloudTrail 在您創建 AWS 帳戶時，您的帳戶已啟用。當活動發生在 AWS 付款密碼編譯中時，該活動會與事件歷史記錄中的其他 AWS 服務 CloudTrail 事件一起記錄在事件中。您可以在帳戶中查看，搜索和下載最近的事 AWS 件。如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

如需 AWS 帳戶中持續記錄事件 (包括 AWS 付款密碼學的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。根據預設，當您在主控台中建立追蹤時，追蹤會套用至所有 AWS 區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔
- 從多個帳戶接收 CloudTrail 日誌文件

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 要求是使用 root 或 AWS Identity and Access Management (IAM) 使用者認證進行。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱CloudTrail userIdentity 元素。

控制平面事件 CloudTrail

CloudTrail 記錄 AWS 付款密碼編譯作業，例如CreateKey、ImportKey、DeleteKeyListKeysTagResource、和所有其他控制平面作業。

資料事件 CloudTrail

資料事件提供在資源上或在資源中執行的資源作業的相關資訊，例如加密承載或轉譯 PIN 碼。資料事件是預設 CloudTrail 不會記錄的大量活動。您可以使用或控制台為 AWS 付款加密資料通道事件啟用 CloudTrail APIs資料事件API動作記錄。如需詳細資訊，請參閱《AWS CloudTrail 使用者指南》中的記錄資料事件。

使用時 CloudTrail，您必須使用進階事件選取器來決定要記錄和記錄哪些 AWS 付款密碼編譯 API活動。若要記錄 AWS 付款密碼編譯資料通道事件，您必須包含資源類型和。`AWS Payment Cryptography key` `AWS Payment Cryptography alias`設定此選項後，您可以藉由選取要記錄的特定資料事件 (例如使用 `eventName` 篩選條件追蹤 `EncryptData` 事件)，進一步調整記錄偏好設定。如需詳細資訊，請參閱〈AWS CloudTrail API參考〉`AdvancedEventSelector`中的〈〉。

> **Note**
>
> 若要訂閱 AWS 付款密碼編譯資料事件，您必須使用進階事件選取器。我們建議您訂閱金鑰和別名事件，以確保您收到所有事件。

數據事件：

- `DecryptData`
- `EncryptData`
- `GenerateCardValidationData`
- `GenerateMac`
- `GeneratePinData`
- `ReEncryptData`
- `TranslatePinData`
- `VerifyAuthRequestCryptogram`
- `VerifyCardValidationData`
- `VerifyMac`
- `VerifyPinData`

資料事件需支付額外的費用。如需詳細資訊，請參閱AWS CloudTrail定價。

瞭解 AWS 付款密碼編譯控制平面記錄檔項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共API調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範「 AWS 付款密碼編譯」`CreateKey` 動作的 CloudTrail 記錄項目。

```
      {
    CloudTrailEvent: {
      tlsDetails= {
        TlsDetails: {
          cipherSuite=TLS_AES_128_GCM_SHA256,
          tlsVersion=TLSv1.3,
          clientProvidedHostHeader=controlplane.paymentcryptography.us-
west-2.amazonaws.com
        }
      },
    requestParameters=CreateKeyInput (
      keyAttributes=KeyAttributes(
        KeyUsage=TR31_B0_BASE_DERIVATION_KEY,
        keyClass=SYMMETRIC_KEY,
        keyAlgorithm=AES_128,
        keyModesOfUse=KeyModesOfUse(
          encrypt=false,
          decrypt=false,
          wrap=false
          unwrap=false,
          generate=false,
          sign=false,
          verify=false,
          deriveKey=true,
          noRestrictions=false)
        ),
      keyCheckValueAlgorithm=null,
      exportable=true,
      enabled=true,
      tags=null),
    eventName=CreateKey,
    userAgent=Coral/Apache-HttpClient5,
    responseElements=CreateKeyOutput(
```

```
      key=Key(
        keyArn=arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp,
          keyAttributes=KeyAttributes(
            KeyUsage=TR31_B0_BASE_DERIVATION_KEY,
            keyClass=SYMMETRIC_KEY,
            keyAlgorithm=AES_128,
            keyModesOfUse=KeyModesOfUse(
              encrypt=false,
              decrypt=false,
              wrap=false,
              unwrap=false,
              generate=false,
              sign=false,
              verify=false,
              deriveKey=true,
              noRestrictions=false)
            ),
          keyCheckValue=FE23D3,
          keyCheckValueAlgorithm=ANSI_X9_24,
          enabled=true,
          exportable=true,
          keyState=CREATE_COMPLETE,
          keyOrigin=AWS_PAYMENT_CRYPTOGRAPHY,
          createTimestamp=Sun May 21 18:58:32 UTC 2023,
          usageStartTimestamp=Sun May 21 18:58:32 UTC 2023,
          usageStopTimestamp=null,
          deletePendingTimestamp=null,
          deleteTimestamp=null)
        ),
      sourceIPAddress=192.158.1.38,
        userIdentity={
          UserIdentity: {
            arn=arn:aws:sts::111122223333:assumed-role/TestAssumeRole-us-west-2/
ControlPlane-IntegTest-68211a2a-3e9d-42b7-86ac-c682520e0410,
            invokedBy=null,
            accessKeyId=TESTXECZ5U2ZULLHHMJG,
            type=AssumedRole,
            sessionContext={
              SessionContext: {
                sessionIssuer={
                  SessionIssuer: {arn=arn:aws:iam::111122223333:role/TestAssumeRole-us-
west-2,
                  type=Role,
```

```
                  accountId=111122223333,
                  userName=TestAssumeRole-us-west-2,
                  principalId=TESTXECZ5U9M4LGF2N6Y5}
                },
                attributes={
                  SessionContextAttributes: {
                    creationDate=Sun May 21 18:58:31 UTC 2023,
                    mfaAuthenticated=false
                  }
                },
                webIdFederationData=null
              }
            },
          username=null,
          principalId=:ControlPlane-User,
          accountId=111122223333,
          identityProvider=null
        }
      },
      eventTime=Sun May 21 18:58:32 UTC 2023,
      managementEvent=true,
      recipientAccountId=111122223333,
      awsRegion=us-west-2,
      requestID=151cdd67-4321-1234-9999-dce10d45c92e,
      eventVersion=1.08, eventType=AwsApiCall,
      readOnly=false,
      eventID=c69e3101-eac2-1b4d-b942-019919ad2faf,
      eventSource=payment-cryptography.amazonaws.com,
      eventCategory=Management,
      additionalEventData={
    }
  }
}
```

瞭解 AWS 付款密碼編譯資料平面記錄檔項目

Dataplane 事件可以選擇配置和功能類似於控制面日誌，但通常容量要高得多。鑑於 AWS 支付加密數據通道操作的某些輸入和輸出的敏感性質，您可能會發現某些字段帶有消息「*** 敏感數據編輯 ***」。這是無法設定的，旨在防止敏感資料出現在記錄檔或追蹤中。

下列範例顯示示範「 AWS 付款密碼編譯」`EncryptData` 動作的 CloudTrail 記錄項目。

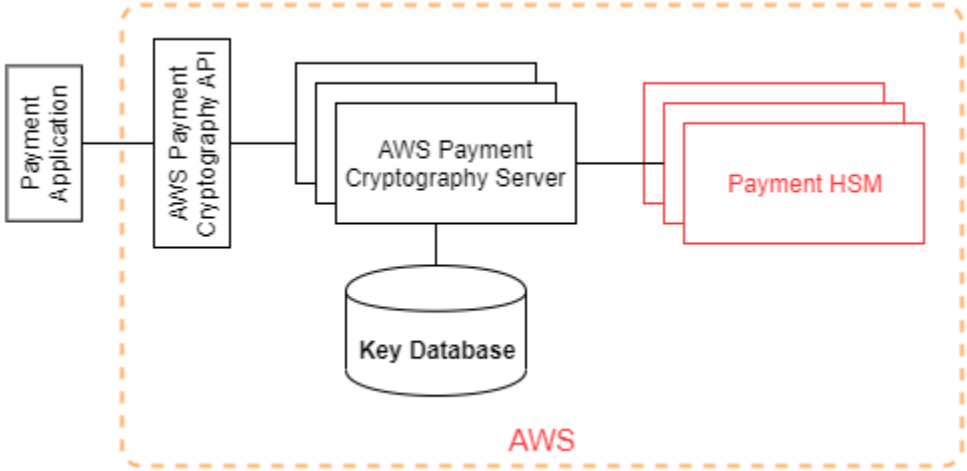
```
      {
        "Records": [
            {
                "eventVersion": "1.09",
                "userIdentity": {
                    "type": "AssumedRole",
                    "principalId": "TESTXECZ5U2ZULLHHMJG:DataPlane-User",
                    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/DataPlane-
User",
                    "accountId": "111122223333",
                    "accessKeyId": "TESTXECZ5U2ZULLHHMJG",
                    "userName": "",
                    "sessionContext": {
                        "sessionIssuer": {
                            "type": "Role",
                            "principalId": "TESTXECZ5U9M4LGF2N6Y5",
                            "arn": "arn:aws:iam::111122223333:role/Admin",
                            "accountId": "111122223333",
                            "userName": "Admin"
                        },
                        "attributes": {
                            "creationDate": "2024-07-09T14:23:05Z",
                            "mfaAuthenticated": "false"
                        }
                    }
                },
                "eventTime": "2024-07-09T14:24:02Z",
                "eventSource": "payment-cryptography.amazonaws.com",
                "eventName": "GenerateCardValidationData",
                "awsRegion": "us-east-2",
                "sourceIPAddress": "192.158.1.38",
                "userAgent": "aws-cli/2.17.6 md/awscrt#0.20.11 ua/2.0 os/macos#23.4.0
 md/arch#x86_64 lang/python#3.11.8 md/pyimpl#CPython cfg/retry-mode#standard md/
installer#exe md/prompt#off md/command#payment-cryptography-data.generate-card-
validation-data",
                "requestParameters": {
                    "key_identifier": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp",
                    "primary_account_number": "*** Sensitive Data Redacted ***",
                    "generation_attributes": {
                        "CardVerificationValue2": {
                            "card_expiry_date": "*** Sensitive Data Redacted ***"
```

```
                        }
                    }
                },
                "responseElements": null,
                "requestID": "f2a99da8-91e2-47a9-b9d2-1706e733991e",
                "eventID": "e4eb3785-ac6a-4589-97a1-babdd3d4dd95",
                "readOnly": true,
                "resources": [
                    {
                        "accountId": "111122223333",
                        "type": "AWS::PaymentCryptography::Key",
                        "ARN": "arn:aws:payment-cryptography:us-
east-2:111122223333:key/5rplquuwozodpwsp"
                    }
                ],
                "eventType": "AwsApiCall",
                "managementEvent": false,
                "recipientAccountId": "111122223333",
                "eventCategory": "Data",
                "tlsDetails": {
                    "tlsVersion": "TLSv1.3",
                    "cipherSuite": "TLS_AES_128_GCM_SHA256",
                    "clientProvidedHostHeader": "dataplane.payment-cryptography.us-
east-2.amazonaws.com"
                }
            }
        ]
    }
```

密碼詳細資料

AWS 支付密碼學提供了一個 Web 界面，用於生成和管理支付交易的加密密鑰。 AWS 付款密碼學提供標準金鑰管理服務和支付交易密碼編譯和工具，可用於集中管理和審計。本文件提供您可在 AWS 付款密碼編譯中使用的密碼編譯作業的詳細說明，以協助您評估服務所提供的功能。

AWS 付款密碼編譯包含多個界面 (包括 RESTful API，透過 AWS CLI、AWS 開發套件和 AWS Management Console)，可針對經 PCI PTS HSM 驗證的硬體安全模組的分散式叢集請求加密操作。



AWS 支付密碼學是一種分層服務，由面向 Web 的 AWS 支付密碼學主機和 HSM 層組成。這些分層主機的分組形成 AWS 支付密碼編譯堆疊。所有對 AWS 付款密碼編譯的要求都必須透過傳輸層安全性通訊協定 (TLS) 提出，並在 AWS 付款密碼編譯主機上終止。服務主機僅允許使用具有提供完美轉發保密的加密套件的 TLS。此服務會使用可用於所有其 AWS 他 API 作業的 IAM 登入資料和政策機制來驗證和授權您的請求。

AWS 付款密碼編譯伺服器會透過私有的非虛擬網路連線至基礎 HSM。服務元件與 HSM 之間的連線會以相互 TLS (MTL) 保護，以進行驗證和加密。

設計目標

AWS付款密碼編譯專為滿足以下需求而設計：

- 可信賴 — 金鑰的使用受到您定義和管理之存取控制政的保護。沒有任何機制來匯出純文字AWS付款金鑰。密碼編譯金鑰的機密性至關重要。若要對 HSM 執行管理動作，需要有多名 Amazon 員工擁有對以仲裁為基礎之存取控制項的角色特定存取權。沒有 Amazon 員工可以存取 HSM 主要 (或主要) 金鑰或備份。主金鑰無法與不屬於AWS付款密碼編譯區域一部分的 HSM 同步處理。所有其他金鑰

均受 HSM 主金鑰保護。因此，客戶AWS付款密碼編譯金鑰無法在客戶帳戶內操作的AWS付款密碼編譯服務之外使用。

- 低延遲和高輸送量 — AWS 付款加密技術提供延遲和吞吐量級別的加密操作，適用於管理付款加密金鑰和處理付款交易。
- 耐久性 — 密碼編譯金鑰的耐久性專門設計為等同於 AWS 中最具耐久性的服務。單一加密金鑰可與支付終端機、EMV 晶片卡或其他使用多年的安全加密裝置 (SCD) 共用。
- 獨立區域 — AWS 為需要限制不同區域，或需要限制不同區域，或需要符合資料駐留需求的客戶提供獨立區域。可以在內隔離金鑰 AWS 量。
- 安全的隨機數字來源 — 由於強大的密碼編譯取決於真正不可預測的隨機數字產生，所以AWS付款密碼編譯取決於真正不可預測的隨機數字產生，所以 AWS支付密碼的所有金鑰產生都使用 PCI PTS HSM 列出的 HSM，並以 PCI 模式運作。
- 稽核 — AWS 付款密碼編譯會在透過 Amazon 取得的日誌和服務CloudTrail日誌中記錄密碼編譯金鑰的使用和管理。CloudWatch您可以使用CloudTrail日誌來檢查密碼編譯金鑰的使用情況，包括包括您擁有共用金鑰的使用情況，包括帳戶。 AWS支付密碼學是由第三方評估機構根據適用的 PCI，卡品牌和地區支付安全標準進行審核。AWS Artifact 提供驗證和共同責任指南。
- 彈性 — AWS 支付密碼學可根據您的需求擴展和擴展。付款密碼學不是預測和保留 HSM 容量，而是按需提供AWS付款密碼編譯。 AWS支付密碼學負責維護 HSM 的安全性和合規性，以提供足夠的容量來滿足客戶的尖峰需求。

基金会

本章中的主題描述了 AWS 支付密碼學的加密原語以及它們的使用位置。他們還介紹了服務的基本要素。

主題

- 每筆交易衍生的唯一金鑰 (DUKPT) 通訊協定
- 金鑰階層

密碼編譯基本元素

AWS 付款密碼學使用可參數化的標準加密演算法，讓應用程式能夠實作其使用案例所需的演算法。這組密碼編譯演算法由 PCI、ANSI X9、EMVCO 和 ISO 標準定義。所有密碼編譯都是由 PCI 模式下執行的 PCI PTS HSM 標準列出的 HSM 執行。

熵和隨機數字產生

AWS 付款密碼編譯金鑰產生會在 AWS 付款密碼學 HSM 上執行。HSM 會實作一個隨機數產生器，以符合所有支援金鑰類型和參數的 PCI PTS HSM 需求。

對稱金鑰作業

支援 ANSI X9 TR 31、ANSI X9.24 和 PCI 引腳附件 C 中定義的對稱式金鑰演算法和關鍵優勢：

- 雜湊函數 — 來自 SHA2 和 SHA3 系列的演算法，輸出大小大於 2551。除了與預 PCI PTS POI v3 終端機的向下相容性以外。
- 加密與解密 — 金鑰大小大於或等於 128 位元的 AES，或金鑰大小大於或等於 112 位元 (2 個金鑰或 3 個金鑰) 的 TDEA。
- 使用 AES 的訊息驗證碼 (MAC) CMAC 或 GMAC，以及具有核准雜湊函式和金鑰大小大於或等於 128 的 HMAC。

AWS 付款密碼編譯會針對 HSM 主金鑰、資料保護金鑰和 TLS 工作階段金鑰使用 AES 256。

注意：一些列出的函數在內部用於支持標準協議和數據結構。如需特定動作支援的演算法，請參閱 API 文件。

非對稱金鑰作業

支援 ANSI X9 TR 31、ANSI X9.24 和 PCI 引腳附件 C 中定義的非對稱金鑰演算法和關鍵優勢：

- 核准的金鑰建立方案 — 如 NIST SP800-56A (以 ECC /FCC2 為基礎的金鑰協定)、NIST SP800-56B (基於 IFC 的金鑰協定) 和 NIST SP800-38F (以 AES 為基礎的金鑰加密/包裝) 中所述。

AWS 付款密碼學主機僅允許使用 TLS 與提供完美前向保密的加密套件連接到服務。

注意：一些列出的函數在內部用於支持標準協議和數據結構。如需特定動作支援的演算法，請參閱 API 文件。

金鑰儲存

AWS 付款密碼編譯金鑰受 HSM AES 256 主要金鑰保護，並儲存在加密資料庫中的 ANSI X9 TR 31 金鑰區塊中。資料庫會複寫到 AWS 付款密碼編譯伺服器上的記憶體內資料庫。

根據 PCI PIN 碼安全性規範附件 C，AES 256 金鑰的強度同等於或強度高於：

- 三鍵 TDEA
- 安全位
- 512 位元
- DSA、衛生署和 MQV 15360/512

使用對稱金鑰匯入金鑰

AWS 付款加密技術支援使用對稱或公開金鑰的密鑰區塊匯入，其對稱金鑰加密金鑰 (KEK) 與受保護的金鑰一樣強或強於匯入的受保護金鑰。

使用非對稱金鑰匯入金鑰

AWS 付款加密技術支援使用對稱或公開金鑰 (KEK) 保護的密碼編譯和金鑰區塊進行匯入，而私密金鑰加密金鑰 (KEK) 與用於匯入的受保護金鑰一樣強或強。提供用於解密的公鑰必須具有其真實性和完整性，由客戶信任的授權機構的證書確保。

AWS 支付密碼學提供的公共 KEK 具有證明符合 PCI PIN 安全性和 PCI P2PE 附件 A 的證明授權單位（CA）的驗證和完整性保護。

金鑰匯出

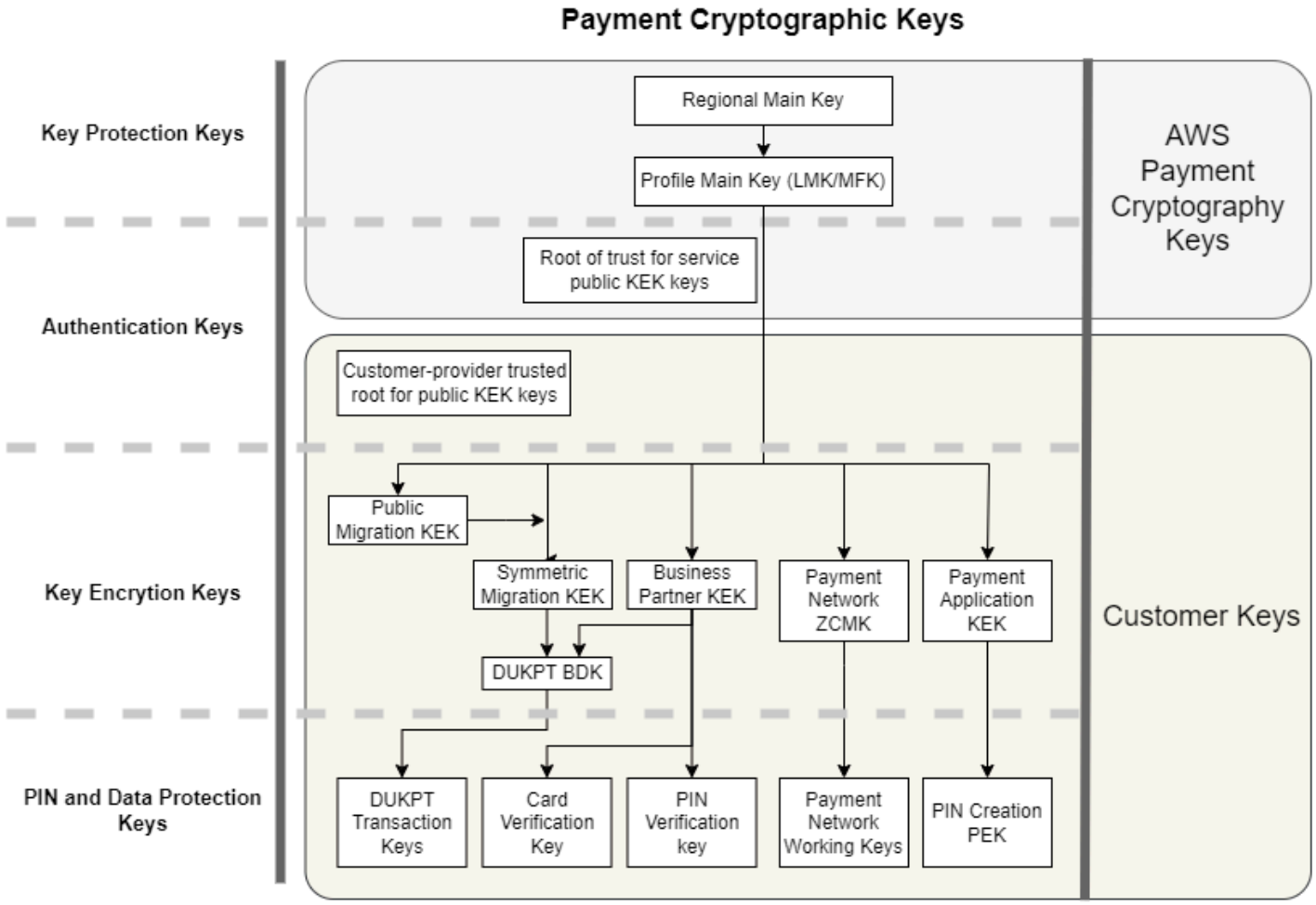
您可以使用適當的金鑰匯出 KeyUsage 和保護金鑰，金鑰強度與要匯出的金鑰一樣強度或強度。

每筆交易衍生的唯一金鑰 (DUKPT) 通訊協定

AWS 支付加密技術支援 TDEA 和 AES 基礎衍生金鑰 (BDK)，如 ANSI X9.24-3 所述。

金鑰階層

AWS 付款密碼編譯金鑰階層可確保金鑰永遠受到與其保護金鑰相同或強度強的金鑰所保護。



AWS 付款密碼編譯金鑰用於服務內的金鑰保護：

金鑰	描述
區域主鍵	保護用於加密處理的虛擬 HSM 映像或設定檔。此金鑰僅存在於 HSM 和安全備份中。
設定檔主鍵	頂層客戶金鑰保護金鑰，傳統上稱為客戶金鑰的本機主要金鑰 (LMK) 或主檔案金鑰 (MFK)。此金鑰僅存在於 HSM 和安全備份中。設定檔會根據付款使用案例的安全性標準所要求，定義不同的 HSM 組態。

金鑰	描述
AWS 付款加密公開金鑰加密金鑰 (KEK) 金鑰的信任根	受信任的根公鑰和證書，用於驗證和驗證由 AWS 付款加密技術提供的公鑰，用於使用非對稱密鑰進行密鑰導入和導出的密鑰。

客戶金鑰會依金鑰分組，用來保護其他金鑰和金鑰，以保護付款相關資料。以下是兩種類型的客戶金鑰範例：

金鑰	描述
客戶為公開 KEK 金鑰提供的受信任根	您作為信任根提供的公開金鑰和憑證，用於驗證和驗證您使用非對稱金鑰匯入和匯出金鑰所提供的公開金鑰。
金鑰加密金鑰 (KEK)	KEK 僅用於加密其他金鑰，以便在外部金鑰存放區和 AWS 付款密碼學、業務合作夥伴、付款網路或組織內的不同應用程式之間進行交換。
每筆交易衍生的唯一金鑰 (DUKPT) 基礎衍生金鑰 (BDK)	BDK 用於為每個支付終端創建唯一的密鑰，並將從多個終端的交易轉換為單個收單銀行或收單機構工作密鑰。PCI 點對點加密 (P2PE) 所要求的最佳做法是，不同的 BDK 用於不同的終端機型號、金鑰插入或初始化服務或其他分段，以限制損害 BDK 的影響。
支付網絡區域控制主密鑰（ZCMK）	ZCMK，也稱為區域密鑰或區域主密鑰，由支付網絡提供，以建立初始工作密鑰。
交易金鑰	為 DUKPT 配置的支付終端機導出終端和交易的唯一密鑰。接收交易的 HSM 可以從終端機識別碼和交易序號決定金鑰。
卡片資料準備金鑰	EMV 發卡機構主金鑰、EMV 卡金鑰和驗證值，以及卡片個人化資料檔案保護金鑰可用於建立個別卡片的資料，以供卡片個人化提供者使用。這

金鑰	描述
	些密鑰和密碼驗證數據也被發卡銀行或發卡機構用於驗證卡數據，作為授權交易的一部分。
卡片資料準備金鑰	EMV 發卡機構主金鑰、EMV 卡金鑰和驗證值，以及卡片個人化資料檔案保護金鑰可用於建立個別卡片的資料，以供卡片個人化提供者使用。這些密鑰和密碼驗證數據也被發行銀行或發卡機構用於驗證卡數據，作為授權交易的一部分。
支付網絡工作密鑰	通常被稱為發行人工作密鑰或獲取者工作密鑰，這些是加密發送到付款網絡或從支付網絡接收的交易的密鑰。這些金鑰會經常由網路輪換，通常是每天或每小時。這些是 PIN 碼/借記卡交易的 PIN 加密密鑰（PEK）。
個人識別號碼 (PIN) 加密金鑰 (PEK)	建立或解密 PIN 區塊的應用程式會使用 PEK 防止儲存或傳輸純文字 PIN 碼。

內部作業

本主題說明此服務實作的內部需求，以保護客戶金鑰和密碼編譯作業，以保護全球分散式且可擴充的付款密碼編譯和金鑰管理服務。

主題

HSM 規格和生命週期

AWS 付款密碼編譯使用市面上可用的 HSM 叢集。HSM 通過 FIPS 140-2 第 3 級驗證，也使用 PCI 安全標準委員會核准的 PCI PTS 裝置清單上列出的韌體版本和安全性原則，作為 PCI HSM v3 投訴。PCI PTS HSM 標準包括 HSM 硬體的製造、出貨、部署、管理和銷毀的額外要求，這些要求對於付款安全性和合規性很重要，但 FIPS 140 無法解決。

所有 HSM 均以 PCI 模式運作，並使用 PCI PTS HSM 安全性原則進行設定。僅啟用支援 AWS 付款密碼編譯使用案例所需的功能。 AWS 付款密碼不提供列印、顯示或退回純文字 PIN 碼。

HSM 裝置實體安全性

服務只能使用製造商在交付前由 AWS 付款密碼編譯憑證授權單位 (CA) 簽署裝置金鑰的 HSM。 AWS 付款密碼編譯是製造商 CA 的子 CA，它是 HSM 製造商和裝置憑證的信任根。製造商的 CA 實作 ANSI TR 34，並已證明符合 PCI PIN 安全性附件 A 和 PCI P2PE 附件 A。製造商會驗證所有含有由 AWS 付款密碼編譯 CA 簽署之裝置金鑰的 HSM 都會運送至 AWS 指定的接收機。

根據 PCI PIN 安全性的要求，製造商會透過與 HSM 貨件不同的通訊管道提供序號清單。在將 HSM 安裝到 AWS 資料中心的每個步驟中，都會檢查這些序號。最後， AWS 付款密碼編譯操作員會根據製造商清單驗證已安裝的 HSM 清單，然後再將序號新增至允許接收 AWS 付款密碼編譯金鑰的 HSM 清單。

HSM 始終處於安全儲存或雙重控制之下，其中包括：

- 從製造商運送到 AWS 機架組裝設施。
- 在機架組裝期間。
- 從機架組裝設施出貨至資料中心。
- 收據並安裝到數據中心安全處理室。HSM 機架透過卡門禁控制鎖、警示門感應器和攝影機，強制執行雙重控制。
- 在操作期間。
- 在退役和銷毀期間。

為每個 HSM 維護和監控完整 chain-of-custody且具有個別責任感。

HSM 初始化

HSM 僅在透過序號、製造商安裝的裝置金鑰和韌體總和檢查碼驗證其身分和完整性後，才會初始化為 AWS 付款密碼學叢集的一部分。驗證 HSM 的真實性和完整性之後，就會進行設定，包括啟用 PCI 模式。然後會建立 AWS 付款密碼編譯區域主金鑰和設定檔主要金鑰，並且 HSM 可供服務使用。

HSM 服務與維修

HSM 具有可維修的元件，不需要違反裝置的加密界限。這些元件包括冷卻風扇、電源供應器和電池。如果 HSM 機架內的 HSM 或其他裝置需要維修，則在機架開啟的整個期間都會維持雙重控制。

HSM 解除委任

因 HSM end-of-life 或故障而發生停用。HSM 在從其機架移除之前 (如果功能正常) 會在邏輯上為零化，然後在 AWS 資料中心的安全處理室中銷毀。它們永遠不會退還給製造商進行維修，用於其他目的，或者在銷毀之前將其從安全的加工室中移除。

HSM 韌體更新

HSM 韌體更新會在必要時套用，以維持與 PCI PTS HSM 和 FIPS 140-2 (或 FIPS 140-3) 列出的版本保持一致性，如果更新與安全性相關，或者判斷客戶可以從新版本的功能中獲益。 AWS 支付密碼學 HSM 會執行 off-the-shelf 韌體，與 PCI PTS HSM 列出的版本相符。新韌體版本已通過 PCI 或 FIPS 認證韌體版本的完整性驗證，然後在推出至所有 HSM 之前測試功能是否具備功能性。

操作員訪問

在極少數情況下，在正常作業期間從 HSM 收集的資訊不足以識別問題或計劃變更，操作員可以非主控台存取 HSM 以進行疑難排解。會執行下列步驟：

- 疑難排解活動已開發並核准，並排程非主控台工作階段。
- HSM 已從客戶處理服務中移除。
- 主鍵被刪除，在雙重控制下。
- 操作員可以非主控台存取 HSM，在雙重控制下執行核准的疑難排解活動。
  - 終止非主控台工作階段後，會在 HSM 上執行初始佈建程序，傳回標準韌體和組態，然後同步主金鑰，然後將 HSM 傳回給服務的客戶。
  - 工作階段的記錄會記錄在變更追蹤中。
  - 從工作階段取得的資訊會用於規劃 future 的變更。

檢閱所有非主控台存取記錄，以確保程序合規性以及 HSM 監控、 non-console-access 管理程序或操作員訓練的潛在變更。

金鑰管理

區域中的所有 HSM 都會同步至區域主要金鑰。區域主鍵可保護至少一個設定檔主要金鑰。設定檔主要金鑰可保護客戶金鑰。

所有主金鑰均由 HSM 產生，並使用非對稱技術透過對稱式金鑰發佈分發給，符合 ANSI X9 TR 34 和 PCI PIN 碼附件 A。

主題

產生

AES 256 位元主要金鑰是使用 PCI PTS 隨機數產生器，在為服務 HSM 叢集佈建的其中一個 HSM 上產生。

區域主鍵同步

HSM 區域主要金鑰會由區域叢集中的服務與 ANSI X9 TR-34 所定義的機制進行同步處理，其中包括：

- 使用密鑰分發主機（KDH）和密鑰接收設備（KRD）密鑰和證書的相互身份驗證，以提供公鑰的身份驗證和完整性。
- 憑證由符合 PCI PIN 附件 A2 要求的憑證授權單位 (CA) 簽署，但非對稱演算法和適用於保護 AES 256 位元金鑰的金鑰優勢除外。

- 與 ANSI X9 TR-34 和 PCI PIN 附件 A1 一致的分散式對稱金鑰的識別和金鑰保護，但非對稱演算法和適用於保護 AES 256 位元金鑰的關鍵優勢除外。

區域主要金鑰可透過下列方式為區域驗證和佈建的 HSM 建立：

- 主要金鑰會在區域中的 HSM 上產生。該 HSM 被指定為金鑰發佈主機。
- 區域中所有佈建的 HSM 都會產生 KRD 驗證 Token，其中包含 HSM 的公開金鑰和不可重複播放的驗證資訊。
- KRD 權杖會在 KDH 驗證 HSM 接收金鑰的身分和權限後，新增至 KDH 允許清單。
- KDH 會為每個 HSM 產生一個可驗證的主金鑰權杖。Token 包含 KDH 驗證資訊和加密的主金鑰，這些資訊只能在為其建立的 HSM 上載入。
- 每個 HSM 都會傳送為其建置的主要金鑰權杖。驗證 HSM 本身的驗證資訊和 KDH 驗證資訊之後，主金鑰會由 KRD 私密金鑰解密並載入到主金鑰中。

如果單一 HSM 必須與區域重新同步處理：

- 它會重新驗證並使用韌體和組態佈建。
- 如果它是該地區的新手：
  - HSM 會產生 KRD 驗證權杖。
  - KDH 會將權杖新增至其允許清單。
  - KDH 會為 HSM 產生一個主要金鑰權杖。
  - HSM 會載入主要金鑰。
  - HSM 可供服務使用。

這可確保：

- 只有在區域內進行 AWS 付款密碼處理驗證的 HSM 才能接收該區域的主金鑰。
- 只有來自 AWS 付款密碼編譯 HSM 的主金鑰可以散發到叢集中的 HSM。

區域主鍵旋轉

區域主要金鑰會在加密期間到期時輪換，在不太可能發生疑似金鑰入侵的情況下，或在確定影響金鑰安全性的服務變更之後進行輪換。

系統會產生新的區域主要金鑰，並與初始佈建一樣散佈。保存的配置文件主鍵必須轉換為新區域主鍵。

區域主要金鑰輪換不會影響客戶處理。

設定檔主要金鑰同步

配置文件主鍵受區域主鍵保護。這會將設定檔限制在特定區域。

設定檔主要金鑰會相應地佈建：

- 設定檔主要金鑰會在已同步區域主金鑰的 HSM 上產生。
- 設定檔主要金鑰會使用設定檔組態和其他內容來儲存和加密。
- 該設定檔可供區域主要金鑰的區域中任何 HSM 用於客戶密碼編譯功能。

輪廓主鍵旋轉

設定檔主要金鑰會在加密期間到期、疑似金鑰遭到入侵之後，或在決定影響金鑰安全性的服務變更之後輪替。

旋轉步驟：

- 新的設定檔主要金鑰會產生並以擱置的主要金鑰的形式散佈，如同初始佈建一樣。
- 背景處理會將客戶金鑰材料從已建立的設定檔主索引鍵轉換為擱置的主金鑰。
- 使用擱置金鑰加密所有客戶金鑰後，擱置金鑰會升級為設定檔主要金鑰。
- 背景程序會刪除受到過期金鑰保護的客戶金鑰材料。

設定檔主要金鑰輪換不會影響客戶處理。

保護

金鑰僅依賴於保護的金鑰階層。保護主金鑰對於防止遺失或損害所有客戶金鑰至關重要。

區域主要金鑰只能從備份還原為服務驗證和佈建的 HSM。這些金鑰只能儲存為特定 HSM 的特定 KDH 可互相驗證的加密主金鑰權杖。

設定檔主要金鑰會以設定檔設定和按區域加密的內容資訊來儲存。

客戶金鑰會儲存在金鑰區塊中，並受到設定檔主要金鑰保護。

所有金鑰僅存在於 HSM 中，或由另一個具有相同或更強密碼編譯強度的金鑰所儲存。

耐久性

即使在通常會導致中斷的極端情況下，也必須提供交易密碼編譯和業務功能的客戶金鑰。 AWS 付款密碼編譯可在可用性區域和區域中使用多層級備援模型。 AWS 客戶需要更高的可用性和耐久性來支付加密操作，而不是服務所提供的實作多區域架構。

HSM 驗證和主要金鑰 Token 會儲存，並可用於還原主金鑰或與新的主金鑰同步處理 (萬一 HSM 必須重設)。令牌被存檔，並在需要時僅在雙重控制下使用。

通訊安全

外部

AWS 付款密碼編譯 API 端點符合 AWS 安全標準，包括 1.2 或更高版本的 TLS 以及用於驗證和請求完整性的簽名版本 4。

傳入 TLS 連線會在網路負載平衡器上終止，並透過內部 TLS 連線轉送至 API 處理常式。

內部 (Internal)

服務元件之間以及服務元件與其他 AWS 服務之間的內部通訊，都受到 TLS 使用強式加密技術的保護。

HSM 位於只能從服務元件存取的私人非虛擬網路上。HSM 與服務元件之間的所有連線均以相互 TLS (MTL) 安全保護，且在 TLS 1.2 或以上。TLS 和 MTL 的內部憑證是由 Amazon Certificate Manager 使用 AWS 私有憑證授權單位來管理。監控內部 VPC 和 HSM 網路是否存在未例外的活動和組態變更。

客戶金鑰管理

和 AWS, 客戶的信任是我們的首要任務. 您可以完全控制 AWS 帳戶下的服務中上傳或建立的金鑰，並負責設定金鑰存取權限。

AWS 付款密碼學對於服務所管理的金鑰的 HSM 實體合規性和金鑰管理負有全部責任。這需要 HSM 主金鑰的擁有權和管理，以及在 AWS 付款密碼編譯金鑰資料庫中儲存受保護的客戶金鑰。

客戶密鑰空間分離

AWS 付款密碼編譯會強制執行所有金鑰使用的金鑰政策，包括將主體限制為擁有金鑰的帳戶，除非金鑰明確與其他帳戶共用。

備份與復原

區域的金鑰和金鑰資訊會由以下方式備份至加密的封存 AWS。歸檔需要雙重控制 AWS 才能還原。

關鍵區塊

所有密鑰都存儲在 ANSI X9 TR-31 格式的鍵塊。

金鑰可以從密碼編譯或支援的其他金鑰區塊格式匯入服務。 ImportKey同樣地，如果金鑰可匯出，也可以匯出為金鑰匯出設定檔支援的其他金鑰區塊格式或密碼編譯。

金鑰使用

金鑰的使用僅限於服務 KeyUsage 所設定。如果要求的密碼編譯作業有不適當的金鑰使用、使用模式或演算法，服務將會失敗任何要求。

關鍵交換關係

PCI PIN 安全性和 PCI P2PE 要求共用加密 PIN 的金鑰 (包括用來共用這些金鑰的 KEK) 的組織，不要與任何其他組織共用這些金鑰。最佳做法是，對稱金鑰只會在兩個當事人之間共用，包括在同一個組織內。如此可將強制取代受影響金鑰的可疑金鑰妥協所造成的影響降到最低。

即使是商業案例，需要共享密鑰超過 2 方之間, 應保持當事人的最小數量.

AWS 付款密碼學提供關鍵標籤，可用於追蹤和強制執行這些需求內的金鑰使用情況。

例如，不同金鑰注入設施的 KEK 和 BDK 可以透過為與該服務提供者共用的所有金鑰設定「KIF」=「POStation」來識別。另一個例子是標記與「網絡」=「PayCard」的支付網絡共享的密鑰。標記可讓您建立存取控制並建立稽核報告，以強制執行並展示您的金鑰管理實務。

金鑰刪除

DeleteKey 在客戶可配置的期限後，標記資料庫中要刪除的金鑰。在此期間之後，金鑰將無法挽回地刪除。這是防止意外或惡意刪除金鑰的安全機制。標記為刪除的金鑰不適用於除外的任何動作 RestoreKey。

刪除後，刪除的金鑰會在服務備份中保留 7 天。在此期間，它們不可恢復。

屬於已關閉 AWS 帳戶的金鑰會標示為要刪除。如果帳戶在刪除期間到達之前重新啟用，則會還原標記為刪除的任何金鑰，但已停用。您必須重新啟用它們，才能將其用於密碼編譯作業。

日誌記錄和監控

內部服務記錄包括：

- CloudTrail 服務發出的 AWS 服務呼叫日誌

- CloudWatch 直接記錄至記 CloudWatch 錄檔或 HSM 事件的兩個事件記錄
- 來自 HSM 和服務系統的記錄檔
- 日誌存檔

所有記錄來源都會監視和篩選敏感資訊，包括關於金鑰。系統檢閱記錄檔，以確保其中包含的記錄不包含敏感的客戶資訊。

完成工作角色所需的個人才能存取記錄。

所有日誌都會保留符合 AWS 日誌保留政策。

客戶操作

AWS 支付密碼學對於 PCI 標準下的 HSM 實體合規性負全部責任。此服務也提供安全的金鑰存放區，並確保金鑰只能用於 PCI 標準允許的目的，且您在建立或匯入期間所指定的金鑰。您必須負責設定關鍵屬性和存取權，以利用服務的安全性和合規性功能。

主題

產生金鑰

建立金鑰時，您可以設定服務用來強制使用金鑰的相容性的屬性：

- 演算法和金鑰長度
- 用量
- 可用性和到期

用於基於屬性的訪問控制（ABAC）的標籤用於限制與特定合作夥伴或應用程序一起使用的密鑰也應在創建過程中設置。請務必包含政策，以限制允許刪除或變更標籤的角色。

您應該確定在建立金鑰之前，已設定決定可以使用和管理金鑰之角色的原則。

> **Note**
>
> 這些 CreateKey 命令的 IAM 政策可用於強制執行和展示金鑰產生的雙重控制。

匯入金鑰

匯入金鑰時，強制使用金鑰的屬性是由服務使用金鑰區塊中的密碼編譯繫結資訊來設定。設定基本金鑰內容的機制是使用使用來源 HSM 建立且受共用或非對稱 KEK 保護的金鑰區塊。這符合 PCI PIN 需求，並保留來源應用程式的使用、演算法和金鑰強度。

除了金鑰區塊中的資訊之外，還必須在匯入時建立重要的金鑰屬性、標籤和存取控制原則。

使用密碼編譯匯入金鑰不會從來源應用程式傳輸金鑰屬性。您必須使用此機制適當地設定屬性。

通常使用明文組件交換密鑰，由關鍵託管人傳輸，然後裝載儀式在安全的房間中實施雙重控制。支 AWS 付密碼學不直接支持這一點。API 將匯出具有憑證的公開金鑰，該憑證可由您自己的 HSM 匯入，以匯出服務可匯入的金鑰區塊。允許使用您自己的 HSM 載入純文字元件。

您應該使用金鑰檢查值 (KCV) 來確認匯入的金鑰是否符合來源金鑰。

ImportKey API 上的 IAM 政策可用於強制執行和展示金鑰匯入的雙重控制。

匯出金鑰

與夥伴或內部部署應用程式共用金鑰可能需要匯出金鑰。使用金鑰區塊進行匯出可維護加密金鑰材料的基本金鑰內容。

金鑰標籤可用來限制將金鑰匯出至共用相同標籤和值的 KEK。

AWS 付款密碼學不提供或顯示純文本關鍵組件。這需要主要託管人直接存取 PCI PTS HSM 或 ISO 13491 通過測試的安全加密裝置 (SCD) 以進行顯示或列印。您可以使用 SCD 建立非對稱 KEK 或對稱 KEK，以在雙重控制下進行純文字金鑰元件建立儀式。

金鑰檢查值 (KCV) 應該用來確認目的地 HSM 匯入的來源金鑰是否相符。

刪除金鑰

您可以使用刪除金鑰 API 來排程在設定一段時間後刪除金鑰。在此之前，時間鍵是可恢復的。一旦金鑰被刪除，它們就會從服務中永久移除。

DeleteKey API 上的 IAM 政策可用於強制執行和展示金鑰刪除的雙重控制權。

輪換 金鑰

使用索引鍵別名可以建立或匯入新金鑰，然後修改索引鍵別名以參照新金鑰，藉此實作金鑰旋轉的效果。根據您的管理實踐，舊密鑰將被刪除或禁用。

的配額 AWS Payment Cryptography

對於每個 AWS 服務，您的 AWS 帳戶有預設配額，先前稱為限制。除非另有說明，否則每個配額都是特定地區的。您可以請求提高某些配額，而其他配額無法提高。

名稱	預設	可調整	描述
Aliases	每個受支援的區域：2,000	是	在當前區域中，您可以在此帳戶中擁有的最大別名數。
控制平面請求的綜合速率	每個支援的區域：每秒 5 個	是	在目前區域中，您可以在此帳戶中每秒發出的控制平面要求數目上限。此配額適用於所有組合的控制平面作業。
資料平面要求的綜合速率 (非對稱)	每個支持的地區：每秒 20 個	是	使用非對稱金鑰的資料平面作業每秒要求數目上限 (您可以在目前區域中的此帳戶中發出)。此配額適用於所有合併的資料平面作業。
資料平面要求的綜合速率 (對稱)	每個支援的區域：每秒 500	是	使用對稱金鑰的資料平面作業每秒要求數目上限 (您可以在目前區域中的此帳戶中發出)。此配額適用於所有合併的資料平面作業。
鍵	每個受支援的區域：2,000	是	當前區域中，您可以在此帳戶中擁有的最大密鑰數量，不包括已刪除的密鑰。

AWS 付款密碼學使用者指南的文件歷史記錄

下表說明 AWS 付款密碼編譯的文件版本。

變更	描述	日期
新區域推出	在歐洲 (法蘭克福)、歐洲 (愛爾蘭)、亞太區域 (新加坡) 和亞太區域 (東京) 新增區域推出端點	2024年7月31 日
CloudTrail 用於數據通道和動態鍵	已新增有關利用 CloudTrail 資料通道 (密碼編譯) 作業的資訊，包括範例。也新增了有關在某些功能中使用動態金鑰，以更好地支援不應匯入到 AWS 付款密碼學的一次性或有限使用金鑰的資訊	2024年7月10日
更新的例子	新增發卡片的新範例	2024年7月1日
功能版本	在VPC端點（PrivateLink）和 iCVV 示例上添加信息。	2024年5月30日
功能版本	新增有關使用RSA和匯出DUKPT IPEK /IK 金鑰的金鑰匯入/匯出新功能的資訊。	2024年1月15日
初始版本	AWS 支付密碼學用戶指南的初始版本	2023 年 6 月 8 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。